Computer Aid for Double Beam Designs

PROJECT REPORT

P-23

Submitted by

VIJAYARAGAVAN S.

SUDHA P.

SASIKALA N.

Under the guidance of

Mr. S. GOPALAN, B.E., M.S.

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF
BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING
OF THE BHARATHIAR UNIVERSITY

1995 - 96

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Kumaraguru College of Technology

COIMBATORE - 641 006

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Kumaraguru College of Technology

COIMBATORE - 641 006

CERTIFICATE

This is to Certify that the report titled

COMPUTER AID FOR DOUBLE BEAM DESIGNS

has been submitted by

Mr. / Ms. [illegible]

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING IN THE COMPUTER SCIENCE AND ENGINEERING BRANCH

of the Bharathiar University, Coimbatore-641 046 during the academic year 1995 - 96.

[illegible]

Guide

Head of the Department

Certified that the candidate was examined by us in the project work Viva-voce held on [illegible] and the University Register Number is [illegible]

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

This project was realized only through the beneficence of many. We would like to thank them all.

We take this opportunity to paean **Dr.S.SUBRAMANIAN,M.Sc.,(Engg.) , Ph.D., S.M.IEEE., M.I.S.T.E.,** Principal, Kumaraguru College of Technology, for his benediction, which has what made us work on our project with more fervor.

We are indebted forever to our venerable Prof. **P.SHANMUGAM,M.Sc(Engg.),M.S(Hawaii),SMIEEE,MISTE,** Head of Department of Computer Science & Engineering and our beloved Project mentor, **Mr.Gopalan,** B.E.,M.S., for their wonderful guidance and benevolent suggestions.

We find pleasure in thanking **MR.HARISOMASUNDARAM, M.Engg, MC(CANADA)**, without whose fecund and timely views we would have been unable to complete our project with finesse.

Our acknowledgement would be incomplete without thanking our serion of friend **Mr. P.Senthil Kumar B.E** , who has helped at all juncture.

We would also like to express our gratitude to our teaching, non-teaching staffs,friends and juniors of our department.

We also thank the proprietor of Arthanari Loom Center who took time from his busy schedule, to extend his help.

SYNOPSIS

The emergence of computers in the Textile, has upto date been the best advancement in this area. All the laborious tasks like card cutting, dyeing etc have been simplified.

Previously, for double warped designs, the designer spent a considerable amount of time, a week or 2, to draw the design, split them according to their colours and other complicated process. From this, the card cutting was done. The whole weaving process was delayed by this.

Our project aims at solving this Problem; The user is given an option of either selecting a picture and scanning it and then modify it; else he can draw the image himself; whether the image is scanned or drawn, everything is done in a grid; the images can also be coloured. That image is then split on grouped into their different colors; this is exactly what is needed at the moment.

Using this split image copy, the card cutting process can be done very easily. This project will thus reduce the timing to a matter of a few minutes. This is a great improvement in performance timing.

CONTENTS

1. INTRODUCTION

Textiles have an important bearing on our daily lives. From time immemorial, people have used textiles of various types for covering, warmth,and personal wealth. Today, textiles are still used for these purposes and everyone is an ultimate consumer. Thus study of textile will be serviceable for specific purposes.

Textile denotes fibers that can be spun into a fabric by interlacing in a variety of methods, including weaving, knitting, braiding, felting and twisting. The fibers that have been woven or are capable of being woven or knitted or bonded into cloth are 'textile' fibers.

"Textile" includes all clothing's, carpets, quilts, upholstery, curtains, etc. Besides wool, silk, linen and cotton it also includes various man-made fibers which may be converted into cloth by conventional or unconventional method. Textile technology denotes the arts, sciences of fibers and fabrics including spinning weaving, knitting, etc., and their mechanism, construction, description.

2. ALL ABOUT WEAVING

2.1 CLASSIFICATION OF WOVEN FABRICS :

Woven fabrics are composed of longitudinal or warp threads and transverse or weft threads, interlaced with one another according to the class of structure and form of design that are desired. The terms chain or twist are applied to the warp, and the warp threads are known individually as ends, while the terms picks and filling are applied to the weft threads. In the following, the term threads is used in referring to warp and weft collectively, but in order to distinguish clearly one series from the other the warp threads are mostly described as 'ends', and the weft threads as 'picks'.

Woven structures may be conveniently divided into two principal categories, as follows:

* Simple structures, in which the ends and the picks intersect one another at right angles and in the cloth are respectively parallel with each other. In these constructions there is only one series of ends and one series of picks and all the

constituent threads are equally responsible for both the aspect of utility or performance in a fabric and the aspect of aesthetic appeal.

* Compound structures, in which there may be more than one series of ends or picks some of which may be responsible for the 'body' of the fabric, such as ground yarns, whilst some may be employed entirely for ornamental purposes such as 'figuring' or 'face' yarns. In these clothes some threads may be found out to be in parallel formation one to another in either plane, and indeed, there are many pile surface construction in which some threads may project out at right angles to the general plane of the fabric.

2.2 TYPES OF WEAVE :

Plain weave, which repeats on two warp and two weft threads, is the simplest and the most frequently used weave. Warp rib, weft rib and matt are weaves derived from plain weave.

Twill weaves are characterized by diagonal lines in the fabric and can be produced with minimum three threads in a repeat. Weaves such as cavalry twill, herring bone twill, pinted twill, diamond designs are derived from basic twill weaves.

Sateen and Satin weaves, most commonly repeating on five to eight threads are characterized by smooth feel and lustrous appearance.

Honeycomb, huckaback, mockleno and leno weaves are cellular structures, with hollows or holes. Crepe weave gives cloth an irregular surface due to random distribution of weave floats. Double cloth structure consists of two separate fabric woven at the same time on one loom but held together by interchanging ends of one fabric weaving with picks of the other.

2.3 CLASSIFICATION OF WEAVING MACHINES :

Looms and weaving machines fall into a number of categories, which differ considerably in the actual methods of weft insertion. The classification can be made as,

(1) Shuttle Looms:

* hand looms
* power looms
* automatic looms

a. bobbin change looms

b. shuttle change looms

(2) Shuttleless Machines:

* Gripper machines
* Rapier machines
* Jet machines
* Multiphase machines

2.3.1 BASIC MECHANISMS OF A PLAIN LOOM:

The loom is a machine made up of several mechanisms, motions and devices which are timed and are required to operate in a proper sequence.

2.3.2 CLASSIFICATION OF MOTIONS :

The motions of a plain loom can be classified as follows:

(I) **Primary motions** :

(a) Shedding (b) Picking (c) beating

(II) **Secondary motions** :

(a) let-off (b) take-up (c) warp protection

(d) weft fork (e) brake (f) temples.

(III) Auxiliary motions :

(a) top rollers (b) box swell (c) check strap

(d) buffer (e) oscillating backrest.

Shedding, picking, beating, let-off and take-up motions are the basic mechanisms necessary for weaving and without which cloth cannot be made. Warp protector motion either loose reed type or fast need type weft fonk motion, brade motion and templer are necessary for successful weaving. Auxiliary motion or devices assist or modify primary motions.

Shedding :

Object of shedding :

Shedding motion divides warp sheet into two layers to form a shed for easy passage of the shuttle, which lays a pick of weft from one side of the sley to the other.

Healds:

Healds are necessary for formation of a shed, when a large number of warp threads are required to move together in the same sequence. The number of heald shafts, to be used depends upon the warp repeat and a limit is reached from 20 to 24 shafts, beyond which it is impossible as the space between the front back heald is not sufficient.

TYPES OF SHEDDING:

Shedding motions are of three types:

(a) Tappet shedding

(b) Dobby shedding

(c) Jacquard shedding

(a) Tappet shedding is the simplest type of motion driven from bottom shaft and is suitable for weaves repeating on two picks such as plain or weft rib and weave repeat can be extended upto six threads for weaves like twills, matts and sateens by mounting suitable shedding tapper on an auxiliary tappet shaft. It is practicable to use upto eight or ten shedding tappets and the number of picks per weave repeat equals the number of shedding tappets used.

(b) Dobby shedding is suitable of weaving patterns which are beyond the range of shedding tappets, either in the number of heald shafts or picks per pattern repeat. Dobbies usually control upto 16 and sometimes as many as 24 heald shafts. There are dobbies with capacity of forty jacks, but they work with harness mail mounting instead of healds. There is no theoretical limit to the number of picks per repeat in dobby shedding.

(c) Jacquard shedding is used for weaving large pattern repeats and each is controlled independently through a harness. The common size of jacquards are 100s, 200s, 400s, and 600s capacity and there are jacquards having capacity beyond 1600.

Picking :

Object of picking:

Picking is next to shedding in the sequence of primary motions. It consists of passing weft yarn through the opened shed of warp threads. The main functions of any conventional picking, where a fly shuttle is used as weft carrier are: (a) delivering shuttle along the correct path for its flight and (b) projection shuttle at a predetermined velocity.

Beating:

Object of Beating:

Beating up follows picking and beats pick of weft left in the shed, to the fell of cloth. Besides, the sley performs two other functions; it houses and acts as raceboard for shuttle and it spaces the ends for obtaining number of ends per inch.

3. NEED FOR CATEX

3.1 EXTRA WARP FIGURING

The chief advantage of the extra warp method is in productivity but its mostly utilized for continuous styles arranged one end of ground, one of extra, except dobby effects which are still produced in a considerable variety of intermittent figuring arrangements. Additional costs are incurred by the net to draw-in new warps into newly re-tied harness which is more expensive than knotting-in , a procedure used when standard harness sets remains unchanged between two warps. The higher actual weaving cost due to the lower rate of production may be partially or entirely offset by weaker and therefore cheap materials for the figuring yarns.

The above argument does not apply quite so strongly in dobby styles where draft changes only necessitates the change in no of slider wires per heald frame although the economics of the drawing-in of new warps as opposed to knoting-in have to taken into account despite the fact that both can be in dobbies performed by machines.

Binding extra ends between face floats

In the ground of ordinary extra warp figured fabrics, it is usually necessary for the extra threads to be invisible from the face side and they can be floated loosely on the back or if the ground weave is suitable, be bound in between corresponding warp floats.

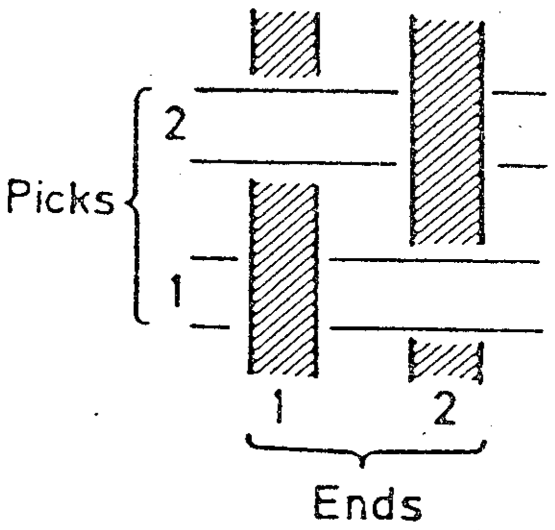
3.2 METHODS OF FABRIC REPRESENTATION:

The unit of woven fabric is the point of intersection of a warp end and a weft pick, the interlacing being of two possible kinds. In either case, the interlacing is achieved by the manipulation of the ends,

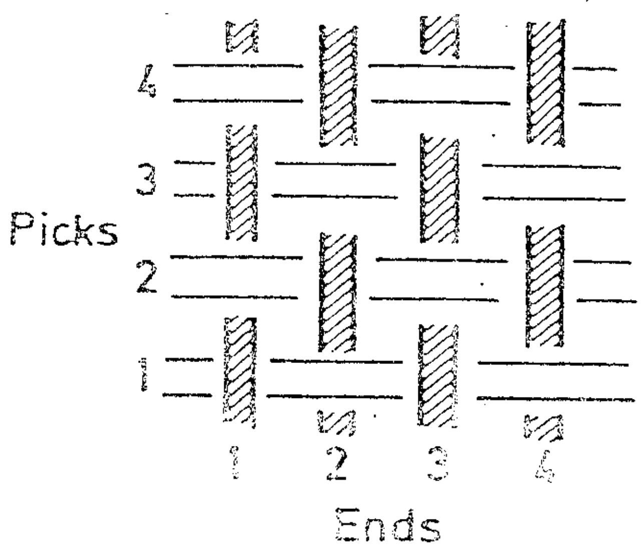
(a) these being raised to obtain the interlacing

(b) or lowered to produce the interlacing.

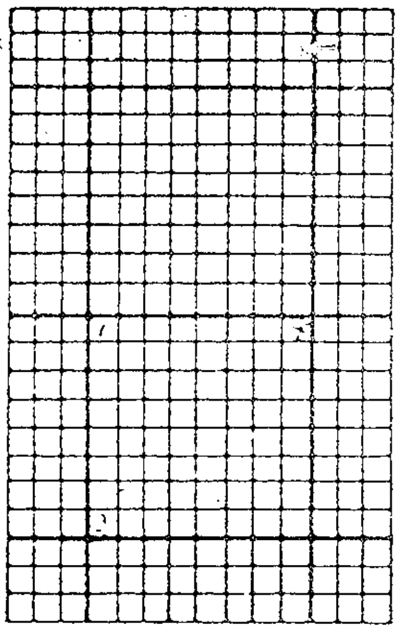
A number of these interlacing combined together in both direction produces a unit of design of one repeat on the weave. Interlacing diagrams [fig 1.4] are not normally employed in designing woven fabrics as they are too laborious to prepare especially when large designs are considered. They are occasion-



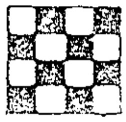
A



B



A



ally used to depict clothes in which threads are dis-placed from the straight path, but in most cases design paper like point paper or squared paper is employed and offers an easy way of representing the inter lacings in a quick and simple manner. The standard textile design paper is ruled in groups of 8 X 8. These being sepa-rated by thicker barlines. [fig 1.5(a)].

Each vertical space is taken to represent a warp end and each horizontal space a weft pick, each square, therefore, indicates an intersection point of an end and a pick.

Square 'X' [fig1.5(b)] indicates the point of crossing of the second end with the second pick, whilst square 'Y' is the point at which the third pick and the fifth end crosses.

Marks are used to denote the interlacing warp over weft and blanks to denote the interlacing warp under weft.

To interlace, the threads must cross one another and therefore in each full repeat of the weave, every vertical space and every horizontal space must have at least one mark and at least one blank otherwise the threads do not interlace but merely form loose floats which don't become woven into the cloth.

3.3 PRINCIPLES OF FIGURING WITH `EXTRA' MATERIALS

A distinguishing feature of fabrics in which extra materials are employed is that the withdrawal of the extra threads from the cloth leaves a complete ground structure under the figure. the formation of a figure by means of extra threads thus does not detract from the strength or wearing quality of the cloth, except that so far as the extra threads are liable to fray out where as in ordinary fabrics ,in which the figure is formed by floating the weft or warp threads loosely the strength of the cloth is reduced-somewhat in proportion to the ratio of figure and ground.

One of the advantages of figuring with extra materials is that the bright colours in sharp contrast with the ground may be brought to the surface of the cloth in any proportion .Pleasing colour combinations may thus be conveniently obtained since the extent of surface allocated to the figuring colour may be readily proportioned in accordance with the degree of its contrast with the ground shade,without the latter being affected.

3.4 METHODS OF INTRODUCING EXTRA FIGURING THREADS

The extra threads may be introduced either as weft or warp or the two methods may be employed in combination. when the extra material is introduced as warp then a separate beam is required for each warp on account of the different take up rates between the extra and the ground ends.For extra weft figuring the weaving machine must have the capacity to insert more than one colour or kind of weft. The form of the design may render it necessary for the extra threads to be inserted in continuous order with the ground threads, or in the intermittent order, while , where they are introduced the arrangement of the figuring ground threads may be 1 and 1, 1 and 2 , 1 and 3, etc accord-ing to the structure of the cloth and the solidity required.

3.5 METHODS OF DISPOSING THE SURPLUS EXTRA THREADS

The disposal of the extra warp or weft threads, in the portions of the cloth where they are not required to form figure, is of great importance,and one or other of the following methods may be employed.

(1) The extra yarn is allowed to float loosely on the back in the ground of the cloth. This is suitable when the space between the figure is not excessive, when the ground is dense and the fabric is used in situations that do not render the long floats on the backs objectionable. It is not applicable to clothes in which the grounds are so light and transparent that the position of the extra thread son the backs can be perceived from the face side.

(2) The extra yarn is allowed to float loosely on the back, and is afterwards cut away. This method is eminently suitable for light ground textures, but if the extra threads float somewhat loosely on the surface in forming the ornament, it is necessary for them to be bounding the edges of the figure, or the loose figuring floats will readily fray out from the surface .The firm interweaving of the extra yarns at the edges, however, makes the outline of the figure less distinct and is rather objectionable unless employed in such a manner as to assist in forming the figure.

(3) In compact fibers the extra threads are bound in on the underside of the cloth, either between corresponding floats in the ground texture or by means of special stitching threads.

(4) The extra threads are interwoven on the face of the cloth in the form of small auxiliary figures or floats thus adding to the fullness of their texture.

3.6 COMPARISON OF EXTRA WARP WITH EXTRA WEFT FIGURING

In extra warp figuring, there are two or more series of warp threads to one series of warp threads, and the method has the following advantages and disadvantages as compared with the extra weft principle:

Advantages:

(1) The productivity of a loom is greater because only one series of picks is inserted, and a faster running loom can be used.

(2) No special picking, box and uptake motions are required.

(3) There is theoretically no limit to the no of colours that can be introduced.

(4) In an intermittent arrangement of the extra ends,either spotted or striped patterns can be formed, whereas a similiar arrangement in the weft can only be used to form spots because of the objectionable appearance of horizontal lines.

Disadvantages:

(1) Two or more warp beams may be required instead of one.

(2) If an ordinary jacquard and harness are employed a smaller width of repeat is produced by a given size of machine because of the set of harness required to be increased in proportion to the no of extra ends that are introduced in a design.

(3) In dobby weaving, the drafts are usually more complicated.

(4) Stronger yarn is required for the figure, and the threads are not so soft and lustrous; extra ends are subjected to greater tension during weaving

than extra picks and as a rule, there is less contraction in length than in width, and the result is that extra warp effects are usually show less prominently than extra warp figures.

(5) If the extra threads have to removed from the underside of the cloth, it is more difficult and costly to cut extra ends than extra picks.

4. COMPUTERSIED WEAVING

4.1 COMPUTERIZED CARD CUTTING:

The latest development in card cutting consists of adapting a computer to process a design from the designer's sketch on squared paper . As the computer can be made to shift a pattern unit, or reflect it about a horizontal or vertical axis, or displace it angularly the artist needs to provide only the basic minimum unit in all symmetrically constructed or repetitive patterns. This unit can be presented in solid painted simplified form and it can be condensed.

Computers and associated systems for this purpose are offered by a number of manufacturers and although the equipment from the various sources differs in some detailed aspects the basic method of operation is similar in each case. A design is painted solid in up to twelve colors to indicate different colour or constructional areas in the cloth and is placed upon a reading table. A photo electronic scanner reading the design is capable of absorbing the information from one complete horizontal row design in, atmost, two seconds. This is transmitted to a control storage unit in an associated computer where it can be modified according to requirements as suggested in the foregoing paragraph.

If the original design was condensed, eg. by four warp-wise and by two weft wise, it will be suitably expanded. Fully worked-out detailed weaves each associated with a different block colour area, are transmitted from a previously assembled 'stock' of weaves stored on a disc to the central control where they are correctly superimposed. Clearly, after a few weeds of operation the stock of detailed weaves will become quite extensive and this can be drawn upon at any time; if, however, new structures, are required they are digitized and simply added to the library as necessary. Full superimposition of all the detailed weaves on all the block colour areas takes place simultaneously for each horizontal row so that a complete card is punched in a single operation.

4.2 SCANNER DETAILS

A color scanner opens up a brilliant range of possibilities for color input and output .Images captured by the scanner can be used to display images on an on-screen. the color scanners normally include image processing and enhancement software for color and gray scale scanning and editing.Scan modules scans images in four different modes:

* 24 bit color images with 16.7 million colors.
* 8 bit grey scale images with 256 shades of gray.
* single-bit black and white line art images.
* single-bit images that simulate greyscale.

The scan module allows software application and hardware imaging devices to communicate directly. After the scan the captured image is automatically placed in the original application. Scanning an image with scan module is a two port process prescanning and scanning. Using the prescan, the scanning area is set. Through actual scanning we can control the resolution, scaling, brightness, contrast and exposure. The scanned image is treated as anew file and should be saved on disk if required later.

The original is scanned with each of the filters inturn, and the results are combined to produce the color image. We have used the color page-I scanner:

The General specifications of a scanner are:

Max Resolution	:	300dpi(H) * 600 dpi(V)
Scanning modes	:	single bit, 8bit, 24 bit color.
Paper Length Selection	:	From 1/8" to 131/2" in 1/8" or 'pixel.
Scanning speed	:	2.8 millisec to 64 millisec / line.
Voltate Req	:	ac 110v to 240v.

Scan Modes:

The digitized images o/p from the scanner contain up to 600 dots per inch of information from the original document. Dots can be represented by a single bit indicating black or white, by an 8 bit code indicating any one of 256 levels of gray, or by a 24 bit code indicating any one of 16 million colors.

Color page-1 scanner has these scanning modes:

Single bit mode: This mode lets you black and white material in line art or half tone mode. Line art is solid black & white, while halftone creates a pattern of black dots.

8-bit Gray mode : This mode uses 8 bits to represent the shading of the original dot with up to 24 bit (8 bit * 3) .This mode reproduces any of 16 million colors .24 bit color is achieved by using a RED,GREEN AND BLUE optical filter in combination with a day light fluorescent lamps.

The basic requirements for installing the "COLOR PAGE SCANNER" on a PC.

* IBM-PC / AT, 386/AT OR 486/AT.
* 2 MB RAM, 40MB hard disk with 5MB free.
* 256 color or 24 bit true color monitor.
* Microsoft Windows 3.1 or above.

4.3 FILE FORMATS

The scanned image is stored in TIFF format.That image is then displayed in a grid, on the user-screen for the user,when requested.The user can then at his wish modify this image using the mouse.Following is a description of the formats of the IMG and TIFF files.

Image file format

Digital research developed the IMG file format for storing pictures as bit images.

The IMG format is simple, consisting of a header followed by pixel information.The information may be encoded in several ways .Encoding compresses the file to occupy less space.

Image data:

IMG encodes images to compress them.There are three ways of encoding pixels : Solid run, Pattern run and Byte string. Solid run encoding describes a sequence that is either all zero's or all ones.

The first 16 bytes are the image header. They are

WORD	BYTE NO	DESCRIPTION
0-1	0-1	Fixed value
2	2-3	Fixed value
2	4-5	Fixed value
3	6-7	Pattern match length
4	8-9	Pixel length in microns, default = 55h
5	0Ah-0Bh	Pixel height in microns, default = 55h
6	0Ch-0Dh	Scan line width in pixels
7	0Eh-0Fh	Number of scan lines in file

Format of a TIFF file :

The TAG IMG FILE FORMAT [TIFF] is a popular method for storing and exchanging digital images.The TIFF format was designed to be portable across any machine architecture.There are two levels of compatibility with the TIFF specification:

Baseline TIFF and TIFF extension.All TIFF files should be totally compatible with the Baseline level. A TIFF file has three components:

* AN IMAGE FILE HEADER
* AN IMAGE FILE DIRECTORY [IFD]
* THE IMAGE DATA itself

A Base line TIFF image is a two dimensional array of pixels, each consisting of color components. Monochromatic data has one component per pixel . whereas RGB color data has three.

TIFF image file header

Byte no.	0	1	2	3	4	5	6	7
Header.	49	49	24	00	08	00	00	00
Contents	byte order		TIFF id		offset of first IFD			

4.4 ELEMENTS OF COLOUR

Luster and color are 2 associated physical phenomenon which demand particular attention from the textile designer due to their prominent influence on the appearance of woven fabrics. When light falls on a fabric some of it maybe reflected at the surface of fibers. Sometimes passing through' one or more fibers before being so reflected and some may be reflected by irregularities within the fibers .The former reflection may be more or less regular as if from a mirror and gives rise to luster. The latter is diffuse reduced luster and if the fabric is dyed giving rise to color.The lustrous appearance of the fabric can depend upon

(a) the characteristics of the fibers.

(b) the way in which the fibers are arranged in the yarn

(c) the weave

(d) the finishing technique applied.

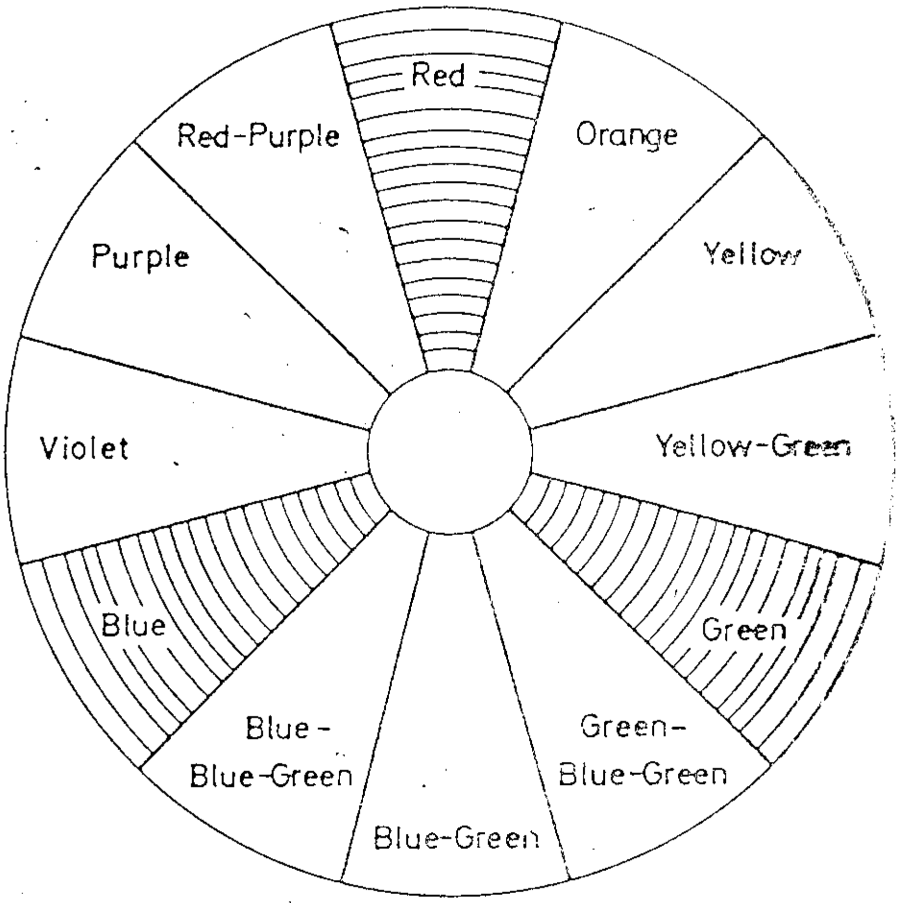
The primary colors are red green and blue .Any color including white can be produced by adding to any color a mixture of the three primaries in right proportion.The mixture of primaries will of course be a color in its own right and it is said to be complimentary to the first color. Thus blue, yellow, green, purple, red and bluish green are complimentary. Complimentary colors are in the greatest possible contrast to one another.

The chromatic circle

Any two complimentary colors are in the greatest contrast to one another.A chromatic circle may be made which enables the colors that are complimentary to be readily seen. The circle is divided into a convenient no of equal parts in this case 12 and at equal distances from each other .The primary colors red blue and green are then painted in.From red to green the colors are then changed through orange ,yellow and yellow green;from green to blue through greenish blue to bluish green and from the blue to the red through violet, purple and reddish purple .Opposite colors in the circle are complementary and in extreme contrast to one another.

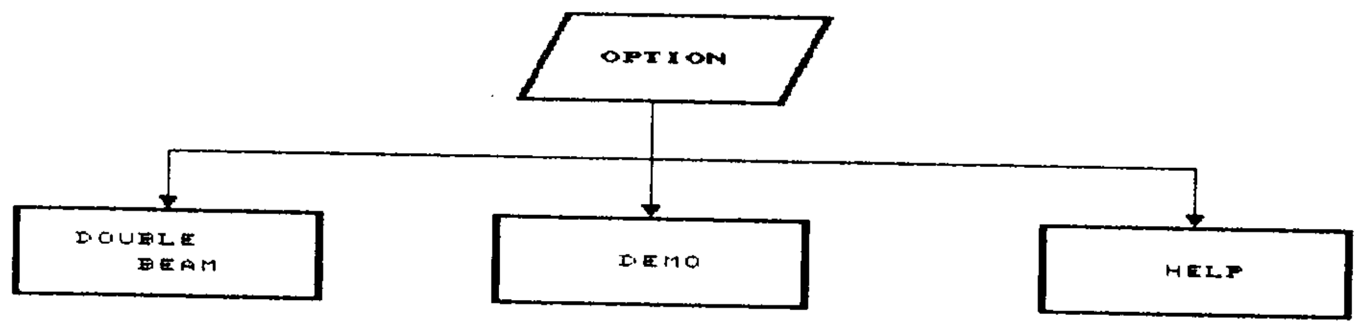
Color combination in relation to weave :

The weaves that are employed in conjunction with combinations of colored threads may be broadly divided into the following 3 classes:


Red
Orange
Yellow
Yellow-Green
Green
Green-Blue-Green
Blue-Green
Blue-Blue-Green
Blue
Violet
Purple
Red-Purple

(1) Weaves that bring the warp and weft threads equally or nearly equal to the surface of the cloth,and enable the colors to be applied in both warp and weft.

(2) Warp face weaves in which the weft is almost entirely concealed so that it is necessary to apply the colors chiefly in the warp.(3) weft face weaves in which the warp is nearly concealed and in which it is seldom possible to apply the color except in the weft.



MAIN BLOCK DIAGRAM

5. SOFTWARE IMPLEMENTATION

EXPLANATION:

The main block has the following modules:

1. Double
2. Demonstration
3. Help.

DOUBLE:

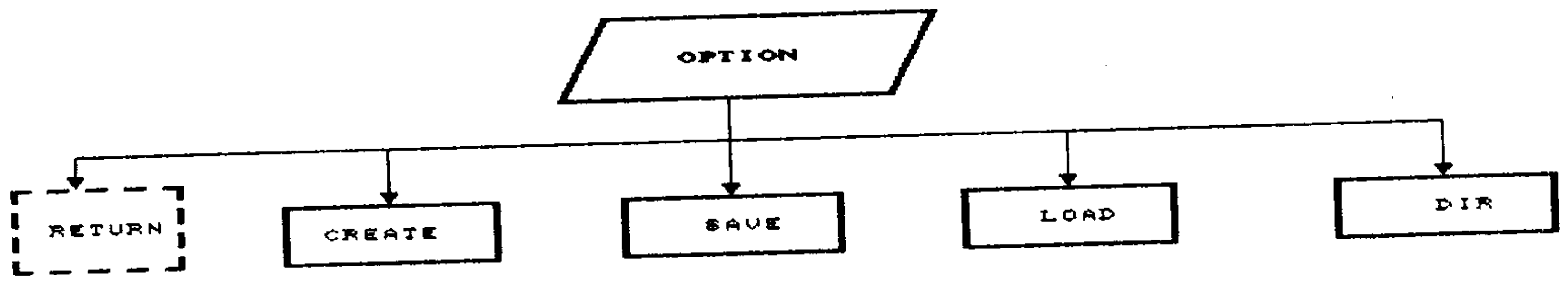
This block contains modules to aid the user to perform various operations on double warp beamed designs. The user can create new designs, alter scanned images, save them or retrieve already stored files. He is also provided with on-line help.

DEMONSTRATION:

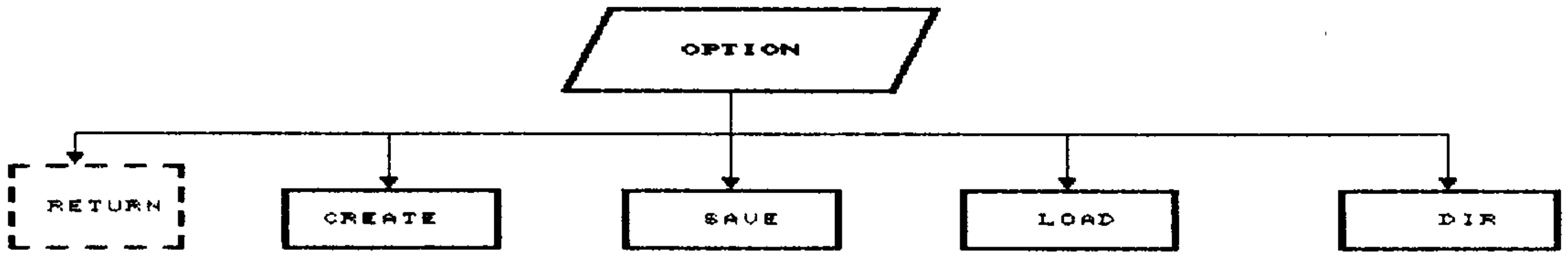
This module displays double warp designs that have been created and saved. further, is also contains a module to display the palette of 256 colour.

HELP:

This module gives a general information about the hardware requirements for this software, definition of the terms used in weaving, textile etc. This helps the user to know, about textile terms used.

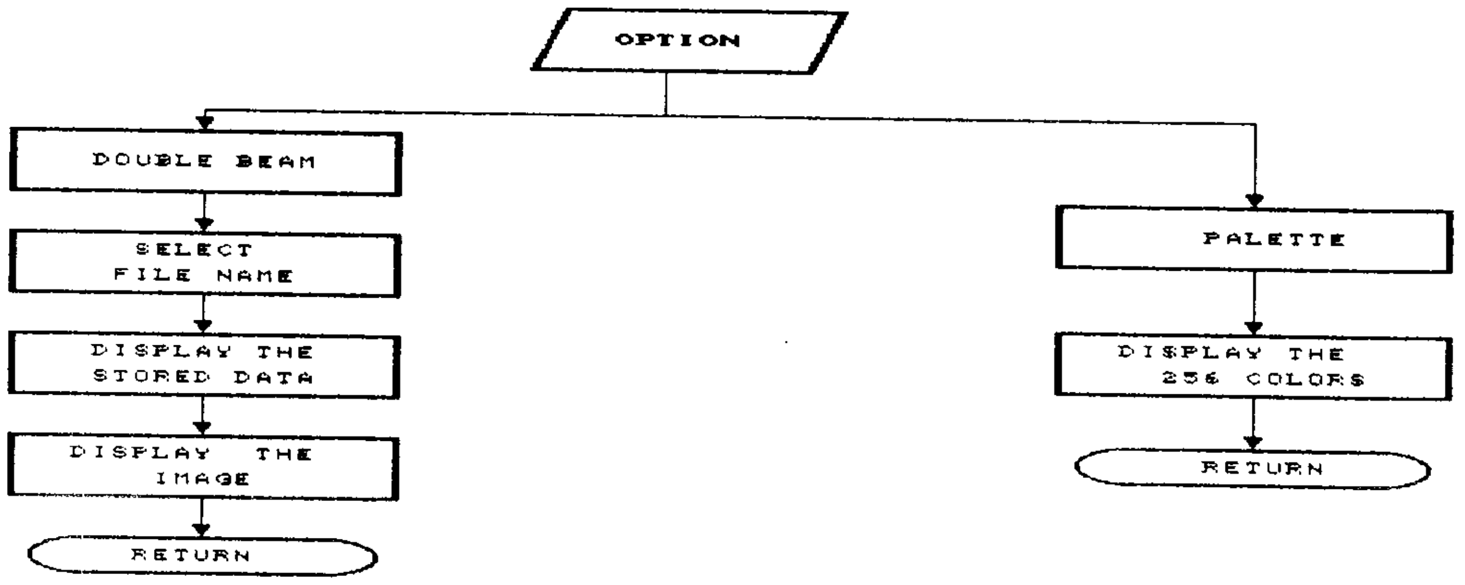


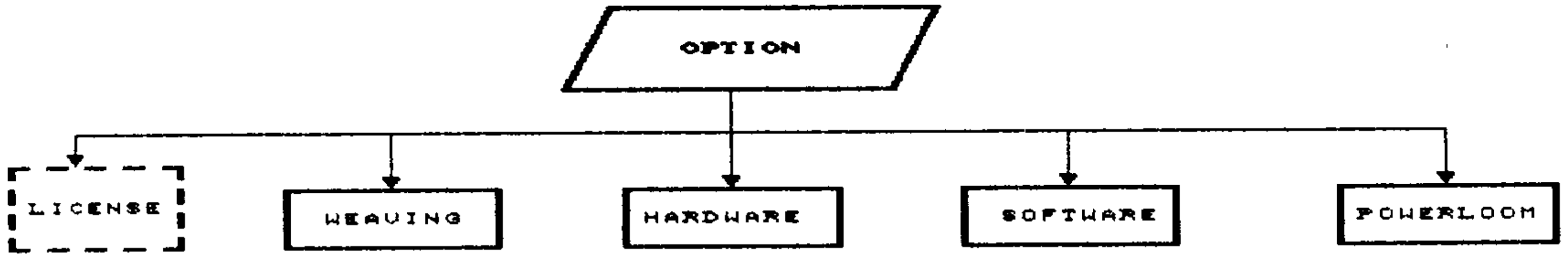
CONTROL KEYS - B.D 1



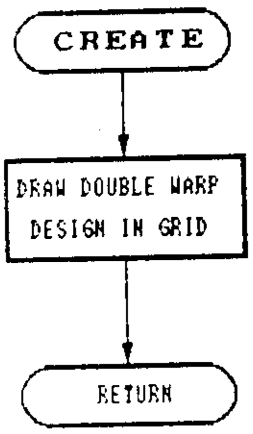
B.D.2 FOR MOUSE

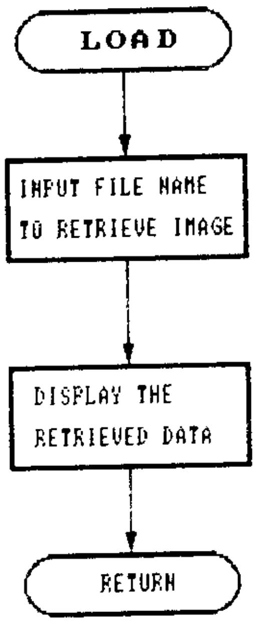
DEMO FLOWCHART

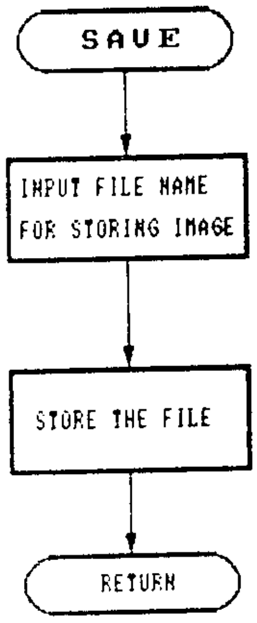


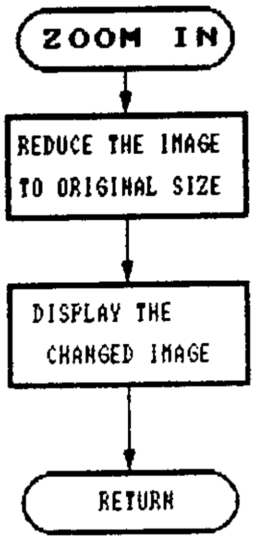


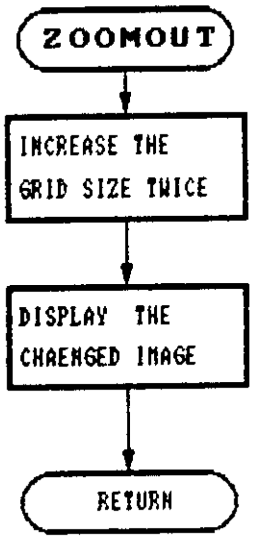
HELP - B.D


CREATE
DRAW DOUBLE WARP DESIGN IN GRID
RETURN


LOAD
INPUT FILE NAME TO RETRIEVE IMAGE
DISPLAY THE RETRIEVED DATA
RETURN


SAVE
INPUT FILE NAME FOR STORING IMAGE
STORE THE FILE
RETURN


ZOOM IN
REDUCE THE IMAGE TO ORIGINAL SIZE
DISPLAY THE CHANGED IMAGE
RETURN


ZOOMOUT
INCREASE THE GRID SIZE TWICE
DISPLAY THE CHAENGED IMAGE
RETURN

6. SOFTWARE DEVELOPMENT

6.1 DOUBLE :

This has the following sub_modules:

6.1.1 CONTROL-KEYS

6.1.2 MOUSE

6.1.3 RETURN

6.1.1 CONTROL KEYS:

CONTROL KEYS HAS THE FOLLOWING MODULES:

(1) **CREATE:** The user will have to select this module if he wishes to create a new design; using the control keys he can draw the design in the grid; he can delete, select a color from the palette.

(2) **SAVE:** To save the design created, the user will have to select this module; they will have to specify a name to store this image.

(3) **LOAD:** To view the previously stored images, the user will have to select this module and specify the name of the file.

(4) DIR: This gives a list of all the images stored.

(5) **HELP:** This is a such- help module, containing informations as to how use these modules.

6.1.2 *MOUSE :*

CREATE : Here the desired image is scanned onto the grid; any modification to be done using the mouse.

The rest of the modules are similar in function as that of the control-key's.

6.1.3 *RETURN :*

Selecting this module takes the action flow back to the main block.

6.2 DEMONSTRATION:

DOUBLE:

This module automatically displays the design, without the user having to specify the datas.

COLOR 256:

This module displays the palette of 256 colors. The user can select a sub-palette of 16 colors for display stile he is creating a design.

8. CONCLUSION

The designs that required double warp beam were simulated on the screen using the control keys, the user creates the whole design; If the design is to be scanned, than that is displayed on the computer screen, which can be altered using the mouse.

The user specifies the width and height of one repeat he then selects colour from the displayed palette. Various options, to load already stored pattern file, to save a newly created pattern have been made available. A library containing some standard pattern and design files was created.

Keeping real time application in minimum, this system has been developed successful to meet specific requirements.

9. FUTURE ENHANCEMENTS

This software supports double beam designs, for jacquard weaving. This can be further enhanced to support multi warp designs. This software can also be enhanced to support more than 256 colors, In this project the user can select 16 colors out of 256 colors only. The resolution mode can further be increased. Yarn counts are not distinguished here. The yarn counts can further be distinguished.

10. REFERENCES

a. Textile Design & Color - Z.J GROSICKI

b. Advanced Textile Design - Z.J GROSICKI
(Newnes - Butterworths Universal Publishing Corporation).

c. C++ programming - ROBERT LAFORE.

d. An Introduction to C++ programming - STEPHEN PRATA.

CATEX CODE

```
# include<iostream.h>
# include<dir.h>
# include<fstream.h>
# include<math.h>
# include<stdio.h>
# include<string.h>
# include<ctype.h>
# include<stdlib.h>
# include<graphics.h>
# include<conio.h>
# include<alloc.h>
# include<dos.h>
# define  PIXEL_SIZE 1
class  menu
{
  private:
          int D_Menu[4][2];
          int Demo[3][2];
          int Help[6][2];
          int C_Keys[6][2];
          int Mouse[6][2];
          int View[4][2];
          int Sub_Help[3][2];
 public:
        void Set_Val();
        void Display(int dis,int index);
        int Mouse_check();
        int Indv_check(int disl,int ind);
        void Disp_Pagel();
        void Disp_Page2();
 };
 int l=0,Press=0,a,b;
 class PROJ_RAIN
 {
   private:
            union REGS regs;
            char *G_Name;
            char *G_by,*D_by;
            char *P_Name;
            char *Pl_Name,*P2_Name,*P3_Name;
            char *a;
   public:
            PROJ_RAIN();
            void Calldisp();
            void Dis(char *temp,int start,int ypos);
            void Cursoroff();
            void Cursoron();
            void Thank_you();
 };
 class Help
 {
   FILE *in;
   char *title;
   int lc; //Line Count
   int length; //Line Length
 char line[80];
 int top;
 int left;
 int right;
 int bottom;
 int cpl; //charecters per line
public:
 Help(char *fname,char *t);
 Help();
 RsHelp();
 void New(char *fname,char *t);
 void New(char *fname,char *t,int x1,int y1,int x2,int y2);
 void Screen();
 void Display();
 void TraceHelp();
 int GetLine();
 void Rewind();
 };
 class Fabric
{
 protected:
  struct palettetype pal;
 public:
  Fabric()
   {
    getpalette(&pal);
   }
   void SetPalette();
   void Palette();
   void RestorePalette();
   void ChangePalette();
 };

void StartGraph();
void Block(int left,int top,int len,int ht,int col,int bcol=17);
void SBlock(int left,int top,int len,int ht,int col,int bcol);
void colorText(int x,int y,char* n,int len,int size,int ForeGround,
                                                 int BackGround);
void Click_Fill(int x,int y,int minx,int miny,int col);
void Input(int x,int y,int&n,int len);
void InputChar(int x,int y,char* n,int len);
void SetBorder(int color);
void ClearViewport(int col);
int GetKey();
void Button_Info();
void ShowMouse();
void IniMouse();
void HideMouse();
void ColorMouse();
extern const DOWN = 20480; extern const UP    = 18432;
extern const  ENTER =   13;extern const  LEFT  =  19200;
extern const  RIGHT =  19712;extern const  PGUP =   18688;
extern const  PGDN =   20736;extern const  HOME =   18176;
extern const  END =    20224;extern const  DELETE =  21248;
extern const TAB  =    9;  extern const  ESC =   27;
extern const  CTRL_F10 =  26368;extern const INSE = 20992;
union REGS in_reg,out_reg;
class Grid
{
 protected:
         int length,width,XMIN,YMIN,XMAX,YMAX;
         char *buff,*color;
         char *unic;
         int *unioccur;
         int ROW,COL,colorno;
         struct palettetype far *pal;
         inline void pointer(int row,int col)
         {
           setcolor(MAXCOLORS);
           setwritemode(XOR_PUT);
           line(XMIN+col*5,YMIN+row*5+3,XMIN+col*5+4,YMIN+row*5+3);
           line(XMIN+col*5+3,YMIN+row*5+1,XMIN+col*5+3,YMIN+row*5+4);
           setwritemode(COPY_PUT);
          }
  public:
          Grid(int XMIN,int YMIN,int XMAX,int YMAX,int ROW,int COL);
          void Set(int x,int y,char *c);
          void Palette();
          void Draw();
          void Nothread_Display();
          void Threadcount();
          void Scale_Img(int Sf_len,int Sf_wth);
          inline void ShadeOne(int x,int y,char color);
          void Shade();
          void Fill();
          void Nothread();
          void Split_Img();
           void Color_Selection();
          void SavePattern();
          void GetPattern();
          void Draw_Img();
          void Cloth_Display();
          void DrawPattern();
          void GetGrid(int x,int y,char *r);
  };
  class Mou_img : public Grid
  {
   protected:
          //int length,width,XMIN,YMIN,XMAX,YMAX,
          int Img_len,Img_wid;
         // char *buff_chg;//,*color;
       //         int ROW,COL;
          char *unic;
          int *unioccur;
 //       struct palettetype far *pal;
   public:
           Mou_img(int XMIN,int YMIN,int XMAX,int YMAX,int ROW,int COL);
           void Set(int x,int y,char *c);
           void Shade();
           void Fill();
          void Draw();
          void Image_Display(char *fnl);
          void DrawPattern(int x,int y);
          void GetGrid(int x,int y,char *r);
          void FillPattern(int x,int y);
          void Threadcount();
          void Nothread_Display();
          void Scale_Img(int Sf_len,int Sf_wth);
          void Nothread();
          void Split_Img();
          void Color_Selection();
          void SavePattern();
          void GetPattern();
          void Draw_Img();
          void Cloth_Display();
 };
 PROJ_RAIN::PROJ_RAIN()
{
  G_Name = (char *)calloc(sizeof(char),25);
  strcpy(G_Name,"Mr. M.GOPALAN B.E.,M.S.");
  G_by = (char *)calloc(sizeof(char),10);
  strcpy(G_by,"GUIDED BY");
  D_by = (char *)calloc(sizeof(char),10);
  strcpy(D_by,"DONE BY");
  P_Name = (char *)calloc(sizeof(char),25);
  strcpy(P_Name,"CATEX DESIGN");
  P1_Name = (char *)calloc(sizeof(char),25);
  strcpy(P1_Name,"S. VIJAYARAGAVAN.");
  P2_Name = (char *)calloc(sizeof(char),25);
  strcpy(P2_Name,"P. SUDHA.");
  P3_Name = (char *)calloc(sizeof(char),25);
  strcpy(P3_Name,"N. SASIKALA.");
  a = (char *)calloc(sizeof(char),25);
}
void Block(int left,int top,int len,int ht,int col,int bcol)
{
  //To set a fill style & color
  if(bcol==17 )
    bcol=col;
  setcolor(bcol);
  setfillstyle(SOLID_FILL,bcol);
  //To set up rectangle coordinates
  int rec[4][2];
  rec[0][0] = rec[3][0] = left;
  rec[0][1] = rec[1][1] = top;
  rec[1][0] = rec[2][0] = left+len;
  rec[2][1] = rec[3][1] = top+ht;
  fillpoly(4,&(rec[0][0]));
}
void PROJ_RAIN::Cursoroff()
{
regs.h.ah = 1;
regs.h.ch = 8;
regs.h.cl = 0;
int86(0x10,&regs,&regs);
```

```
}
void PROJ_RAIN::Thank_you()
  {
    int a,b,i=1;
    do{
    i++;
    outtextxy(100,100,"THANK");
    outtextxy(100,250,"    YOU");
    sleep(1);
    cleardevice();
    }while(i < 5);
  }
void PROJ_RAIN::Calldisp()
{
  Dis(P3_Name,15,19);
  Dis(P2_Name,15,17);
  Dis(P1_Name,15,15);
  Dis(D_by,5,13);
  Dis(G_Name,12,11);
  Dis(G_by,5,8);
}
void PROJ_RAIN::Dis(char *a,int startx,int ypos)
{
   int c,j,i;
   char word;
   int xloc,yloc,count;
   j=0;//int len = strlen(a)/2;
   //int rowval = startx;
   int  t_wd,t_ht,k;
   t_wd = textwidth("W");
   t_ht = textheight("W");
   char st[1],str[1];
   strcpy(str," ");
   for(j=0;j<strlen(a)/2;j++)
   {
    for(int i=0;i<ypos;i++)
    {
     st[0] = *(a+j);
     st[1] = '\0';
     outtextxy((startx+j)*t_wd,i*t_ht,st);
     delay(2);
     st[0] = *(a+(strlen(a)-j-1));
     st[1] = '\0';
     k = strlen(a)-j-1;
     outtextxy((startx+k)*t_wd,i*t_ht,st);
     delay(2);
     Block(0,0,getmaxx(),(ypos-1)*t_ht+3,0);
     setcolor(WHITE);
    }

   }
   if(strlen(a)/2!=0)
     {
     st[0] = *(a+(strlen(a)/2));
    for(int i=0;i<ypos;i++)
```

```
      {
       st[1] = '\0';
       outtextxy((startx+j)*t_wd,i*t_ht,st);
       Block(0,0,getmaxx(),(ypos-1)*t_ht+3,0);
       setcolor(WHITE);
       delay(2);
       }
     }
 }

 void PROJ_RAIN::Cursoron()
 {

 regs.h.ah = 1;
 regs.h.ch = 7;
 regs.h.cl = 8;
 int86(0x10,&regs,&regs);
 }

 /*void main()
 {
   int i,j,k,n,gd=DETECT,gm;
   PROJ_RAIN CRAIN;
   initgraph(&gd,&gm," ");
   clearviewport();
   settextstyle(1,0,3);
   CRAIN.Calldisp();
   getch();
   //CRAIN.Thank_you();
   closegraph();
  }*/


 Help::Help(char *fname,char *t)
 {
  if(((in=fopen(fname,"r"))==NULL))cout<< "file opening error"<<endl ;
  else cout << "success";
  title=t;
  top=100;
  bottom=260;
  left=150;
  right=450;
  lc=0;
  cpl=right-left-25;
  cpl/=8;
 }

 Help::Help()
 {
 top=100;
 bottom=260;
 left=150;
 right=450;
 lc=0;
 cpl=right-left-25;
 cpl/=8;
}

void Help::New(char *fname,char *t,int x1,int y1,int x2,int y2)
{
 if((in=fopen(fname,"r"))==NULL){cout << "error"; exit(0);}
 title=t;
 top=y1;
 bottom=y2;
 left=x1;
 right=x2;
 lc=0;
 cpl=right-left-25;
 cpl/=8;
}

void  Help::New(char *fname,char *t)
{
 if((in=fopen(fname,"r"))==NULL){cout << "error in opening file";exit(0);}
 title=t;
 top=100;
 bottom=260;
 left=150;
 right=450;
 lc=0;
 cpl=right-left-25;
 cpl/=8;
}

 Help::~Help()
{
 fclose(in);
}

void Help::Screen()
{
 SBlock(left,top,right-left,bottom-top,BLUE,DARKGRAY);
 SBlock(left+60,top-30,150,25,BLUE,DARKGRAY);
 setcolor(WHITE);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
 outtextxy(left+70,top-31,title);
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
}

int Help::GetLine()
{
   int j;
   j=-1;
   do
   {
    j++;
    line[j]=fgetc(in);
    while(line[j]=='\t') line[j]=fgetc(in);
    //if(j>cpl) break;
   } while(line[j]!='\n'&&line[j]!=EOF);
```

```
  line[j]='\0';
  lc++;
  length=j;
  if(line[j]==EOF)
   {
    length--;
    return(1);
   }
  return(0);
}

void Help::Display()
{
 int i;
 Block(left+5,top+5,right-left-10,bottom-top-10,GREEN);
 for(i=0;i<12;i++)
 {
   if(!GetLine())
        colorText(left+10,top+10+i*11,line,cpl,1,WHITE,MAGENTA);
   else
     {
      colorText(left+10,top+10+i*11,line,cpl,1,WHITE,MAGENTA);
      sound(250);
      delay(100);
      sound(500);
      delay(100);
      nosound();
      return;
     }
 }
 }

 void Help::Rewind()
 {
  int j=lc;
  lc=0;
  rewind(in);
  for(int i=0;i<j;i++) GetLine();
 }

 void Help::TraceHelp()
 {
  int i;
  Screen();
  Display();
  while(1)
  {
  switch(GetKey())
  {
  case PGDN:

            Display();
            break;
  case DOWN:lc-=11;
            Rewind();
```

```
           Display();
           break;
 case UP   :if(lc>12)lc-=13;
           Rewind();
           Display();
           break;
 case PGUP:lc-=24;
           if(lc<=0)
            lc=0;
           Rewind();
           Display();
           break;
 case ESC: Block(left-10,top-40,right-left+20,bottom-top+50,0);
           return;
 default: continue;
 }
 }
}
/*(void main()
{
 char name[] = "hello.dat";
  StartGraph();
  Help help;
  help.New(name,"   Help");
  help.TraceHelp();
  closegraph();
 } */

void Color256();

void Cone(int x,int y,int col,int dx=10,int dy=100)
{
 int poly[4][2]={{x+dx,y},{x+2*dx,y},{x+3*dx,y+dy},{x,y+dy}};
 setcolor(MAXCOLORS);
 setfillstyle(SOLID_FILL,MAXCOLORS);
 fillpoly(4,&(poly[0][0]));
 poly[0][0]-=1.5*dx;
 poly[3][0]-=1.5*dx;
 poly[1][0]+=1.5*dx;
 poly[2][0]+=1.5*dx;
 poly[0][1]+=(int)(dy/8);
 poly[1][1]+=(int)(dy/8);
 poly[2][1]-=(int)(dy/8);
 poly[3][1]-=(int)(dy/8);
 if(col!=0)
   setcolor(col);
 setfillstyle(SOLID_FILL,col);
 fillpoly(4,&(poly[0][0]));
}
void Fabric::SetPalette()
{
   int col=0,newcol;
   char s[4];
   getpalette(&this->pal);
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
```

```
    for(col=0;col<16;col++)
    {
     Cone(40+(col%8)*70,70+(int)(col/8)*140,col);
         sprintf(s,"%d",col);
     if(col!=MAXCOLORS)
        setcolor(MAXCOLORS);
     else
        setcolor(0);
     outtextxy(50+(col%8)*70,115+(int)(col/8)*140,s);
    }
    ShowMouse();
    setcolor(MAXCOLORS);
    int conx,cony;
    //col=1;
    //newcol=this->pal.colors[col];
    outtextxy(150,15,"SETTING USER PALETTE");
    outtextxy(50,400,"Selected color number is ");
    newcol=(int)this->pal.colors[col];
    sprintf(s,"%d",newcol);
    Block(380,390,100,50,15);
    setcolor(0);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
    outtextxy(390,395,s);
    setcolor(MAXCOLORS);
    Press=0;
    while(1)
    {
    strcpy(s," ");
lo:   conx=cony=col=newcol=0;
     Button_Info();
    conx = (a-40)/70;cony = (b-70)/140;
    col = ((cony*8)+conx);
    newcol=this->pal.colors[col];
    switch(Press)
    {
     case UP : newcol+=8;
               break;
     case DOWN: newcol-=8;
                break;
     case 1:
              if(Press==1)
              {
              newcol--;
              break;
              }
     case 2:
             if(Press==2)
             {
              newcol++;
              break;
             }
     case 4:
              return;
     case ESC:
             getpalette(&this->pal);
             clearviewport();
             return;
     default: break;
    }
     if((Press==0)&&((a>39&&a<620)&&(b>69&&b< 320)))
     {
        conx = (a-40)/70;cony = (b-70)/140;
    col = (cony*8)+conx;
    newcol=(int)this->pal.colors[col];
    if(newcol<0)
       newcol+=256;
    if(newcol>255)
       newcol-=256;
    sprintf(s,"%d",newcol);
    Block(380,390,100,50,15);
    setcolor(0);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
    outtextxy(390,395,s);
    Press=0;
        goto lo;
     }
    if(newcol<0)
       newcol+=256;
    if(newcol>255)
       newcol-=256;
    setpalette(col,newcol);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
    sprintf(s,"%d",newcol);
    Block(80,390,100,50,MAXCOLORS);
    setcolor(0);
    outtextxy(390,395,s);
    setcolor(MAXCOLORS);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    sprintf(s,"%d",col);
    if(col!=MAXCOLORS)
      setcolor(MAXCOLORS);
    else
      setcolor(0);
    outtextxy(50+(col%8)*70,115+(int)(col/8)*140,s);
    setcolor(MAXCOLORS);
    getpalette(&(this->pal));
    Press=0;
  }
 }
     void Fabric::RestorePalette()
     {
      setallpalette(&(pal));
     }
void Fabric::Palette()
{
   int col;
   char s[4];
   settextstyle(SMALL_FONT,HORIZ_DIR,4);
   Block(150,380,300,80,0);
```

```
setcolor(MAXCOLORS);
rectangle(149,379,421,461);
rectangle(151,381,419,459);
for(col=0;col<16;col++)
 {
  Block(155+(col%8)*30,385+(int)(col/8)*40,25,20,ccl);
  sprintf(s,"%d",col);
  setcolor(15);
  outtextxy(160+(col%8)*30,407+(int)(col/8)*40,s);
  //Block(155+(col%8)*30,385+(int)(col/8)*40,25,20,col);
 }
setcolor(MAXCOLORS);
col=0;
rectangle(154+(col%8)*30,384+(int)(col/8)*40,
     181+(col%8)*30,406+(int)(col/8)*40);
setcolor(WHITE);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
}

void Fabric::ChangePalette()
{
 int col=0,newcol;
 char s[4];
 getpalette(&this->pal);
 col=0;
 newcol=this->pal.colors[col];
 Palette();
 outtextxy(430,390,"Selected color");
 Block(440,410,30,30,col,WHITE);
 int op;
 while(1)
 {
 switch(op =GetKey())
 {
  case UP :newcol+=8;
           break;
  case DOWN:newcol-=8;
           break;
  case LEFT:newcol--;
           break;
  case RIGHT:newcol++;
            break;
  case TAB:
            this->pal.colors[col]=newcol;
            setcolor(0);
            rectangle(154+(col%8)*30,384+(int)(col/3)*40,
                      181+(col%8)*30,406+(int)(col/3)*40);
            if(col>=15)
              col=0;
            else
              col++;
           getpalette(&this->pal);
           newcol=this->pal.colors[col];
           setcolor(MAXCOLORS);
            rectangle(154+(col%8)*30,384+(int)(col/8)*40,
                   181+(col%8)*30,406+(int)(col/8)*40);
         Block(440,410,30,30,col,WHITE);
         break;
    /*case SHIFT_TAB:
         setcolor(0);
          rectangle(154+(col%8)*30,384+(int)(col/8)*40,
                   181+(col%8)*30,406+(int)(col/8)*40);
         if(col!=0)
              col--;
         else
            col=MAXCOLORS;
         getpalette(&this->pal);
         newcol=this->pal.colors[col];
         setcolor(MAXCOLORS);
         rectangle(154+(col%8)*30,384+(int)(col/8)*40,
                   181+(col%8)*30,406+(int)(col/8)*40);
         Block(440,410,30,30,col,WHITE);
         break;*/
    case ESC: clearviewport();
              return;
    default: break;
   }
   if(newcol<0)
      newcol+=256;
   if(newcol>255)
      newcol-=256;
   setpalette(col,newcol);
   setcolor(MAXCOLORS);
   getpalette(&(this->pal));
  }
 }
void DosBlock(int x1,int y1,int dx,int dy,int col)
{
 int i,j;
 REGS in;
 for(i=0;i<dx;i++)
  for(j=0;j<dy;j++)
  {
   in.h.ah=0x0c;
   in.h.al=col;
   in.h.bh=0;
   in.x.cx=x1+i;
   in.x.dx=y1+j;
   int86(0x10,&in,&in);
  }
}
void Color256()
{
 REGS in;
 in.h.ah=00;
 in.h.al=0x13;
 int86(0x10,&in,&in);
 int i=0;
 for(i=0;i<256;i++)
  DosBlock((i%20)*15,(int)(i/20)*15,15,15,i);
 getch();
 StartGraph();
}
  void Grid::GetGrid(int x,int y,char *r)
 {
   color=(char *)calloc(sizeof(char),x*y);
   for(int i=0;i<x;i++) color[i]=15;
   Set(x,y,color);
   Draw();Fill();
   for(i=0;i<x*y;i++) r[i] = buff[i];
   return;
 }
 Grid::Grid(int XMIN,int YMIN,int XMAX,int YMAX,int ROW,int COL)
 {
 Grid::ROW = ROW;
 Grid::COL = COL;
 Grid::XMIN = XMIN;
 Grid::YMIN = YMIN;
   Grid::XMAX = XMAX;
 Grid::YMAX = YMAX;
}
void Grid::Set(int x,int y,char *Warp)
{
 length=x;width=y;
 color=Warp;ROW=COL=0,colorno=1;
   XMIN=2;XMAX=XMIN+(length+1)*5;YMIN=108,YMAX=YMIN+width*5;
   buff = (char *) calloc(sizeof(char),(x+1)*y);
   for(x=0;x <= (length+1)*width;x++)
       buff[x]=getbkcolor();
   setpalette(0,7);
   unic = (char *) calloc(sizeof(char),16);
 unioccur = (int *) calloc(sizeof(int),16);
}

void Grid::Draw()
{
 int i,j;
 setcolor(MAXCOLORS);
 for(i=0;i<=width;i++)
   line(XMIN,YMIN+i*5,XMAX,YMIN+i*5);
 for(i=0;i<=(length+1);i++)
   line(XMIN+i*5,YMIN,XMIN+i*5,YMAX);
 SBlock(5,15,length,width,0,MAXCOLORS);
}
void Mou_img::Draw()
{
 int i,j;
 setcolor(MAXCOLORS);
 for(i=0;i<=width;i++)
   line(XMIN,YMIN+i*5,XMAX,YMIN+i*5);
 for(i=0;i<=(length+1);i++)
   line(XMIN+i*5,YMIN,XMIN+i*5,YMAX);
}

void Grid::Nothread()
```

```
        else
          continue;
      }
    }
      if(setcount==1)
      {  setcount=0;break; }
  }
}
return;
}
void Grid::Nothread_Display()
{
   int i;
   char s[2];
   ClearViewport(0);
   SBlock(165,200,400,colorno*13+100,0,MAXCOLORS);
   colorText(200,225,"  COLOR LIST OF CURRENT IMAG ",40,1,0,15);
   for(i=0;i<colorno;i++)
    {
     Block(210,200+i*10+50,40,10,unic[i]);
     char ss = unioccur[i];
     sprintf(s,"%d",ss);
     outtextxy(260,200+i*10+53,s);
     colorText(280,200+i*10+50,"NO. OF THREADS.",30,1,0,15);
    }
    setcolor(MAXCOLORS);
    getch();
    Split_Img();
    getch();
    Cloth_Display();
    return;
}

void  Grid::Split_Img()
{
 int tenp1,tenp2,sro,xcord,n;
 int gaug,temstor=0,cas;
 int tempvalc=0,tempc=0,temval=0,temcount=0;
 double sco;
 int i,l,clear  = 0;
 ClearViewport(0);
 setcolor(15);
  for(l = 0; l < colorno-1;l++)
   {
     tempvalc = 0;
     temcount=0;
    for(i = 0; i < (length+1) ; i++)
     {
      for(n = 0; n<width; n++)
      {
       if(buff[i+n*(length+1)] == unic[l])
        {
                ++temcount;
         double j =i+(length+1)*n;
         sro = floor((i+(length+1)*n)/(length+1)) ;
```

```
          sco = fmod(j,(length+1));
          if((temcount==1)&&(l>=1))
            {
              cas = sco;
              if(sco > temstor)
                 temval = sco + tempc+1;
              else
               if(sco < temstor)
                  temval = tempc+1+(temstor-sco);
            }
          if(sco >= tempvalc)
              tempvalc=sco;
           xcord = XMIN+1+(temval+sco)*10;
           if(xcord == 633)
             {
               setcolor(MAXCOLORS);
               for(i=0;i<=width;i++)
                line(XMIN,YMIN+i*10,XMIN+10*633,YMIN+i*10);
               for(i=0;i<=633;i++)
                 line(XMIN+i*10,YMIN,XMIN+i*10,YMIN+10*width);
              }
          if(xcord > 640)
           {
             ++clear;
             if(clear == 1)
              {
                if(GetKey() == 19712)
                ClearViewport(0);
              }
              xcord = xcord-640;
            }
          Block(xcord,YMIN+1+sro*10,8,8,unic[1]);
       }
    }
 }

       if(l==0)
           temstor = tempvalc;
        if(l>0)
          tempc +=tempvalc+1-cas;
  }
  setcolor(MAXCOLORS);
  gaug = tempc+temstor+1;
  if(gaug < length+1)
    gaug = length+1;
  else
    gaug = gaug - (640/10-XMIN-2);
   for(i=0;i<=width;i++)
   line(XMIN,YMIN+i*10,XMIN+10*gaug,YMIN+i*10);
 for(i=0;i<=gaug;i++)
   line(XMIN+i*10,YMIN,XMIN+i*10,YMIN+10*width);
}


void Grid::Shade()
```

```
{
  int i,j;
  Block(XMIN+135,YMIN-100,XMAX-125,YMIN-25,0);
  Draw();
  Palette();
}
void Grid::Threadcount()
{
 int threadcount=0,oldthcount=0;
 int cval = 0,count=0,color_check[4]={0,0,0,0};
  char s[50];
l:  for(int i=0;i<(length+1);i++)
  {
  oldthcount = threadcount;
  for(int j=0;j<width;j++)
  {
  count = 0;
    if(buff[j*(length+1)+i]!=getbkcolor())
    {
    for(int k=0;k<4;k++)
        if(buff[j*(length+1)+i] == color_check[k])
         {
          if(oldthcount==threadcount)
            ++threadcount;
          break;
         }
        else
          { ++count;
          continue; }
      if(count==4)
      {
        if(cval < 4)
        {
         color_check[cval] = buff[j*(length+1)+i];
         ++cval;
        }
      }
    if(oldthcount!=threadcount)
     break;
  }
  }
}
if(threadcount > length+2)
{
  SBlock(175,25,250,70,0,MAXCOLORS);
    strcpy(s,"NO OF THREAD IS > ");
    colorText(200,33,s,24,1,0,15);
    sprintf(s,"%d",length+1);
    colorText(340,33,s,2,1,0,15);
    strcpy(s,"PLEASE FILL IT WITHIN THIS LIMIT");
    colorText(183,53,s,30,1,0,15);
    strcpy(s,"PRESS ENTER...");
    colorText(225,78,s,20,1,0,15);
    getch();
    Block(165,25,265,81,0);
    Palette();
    threadcount = 0;
    cval=0;
    Fill();
    goto 1;
}
}
void Click_Fill(int x,int y,int minx,int miny,int col)
{
    Block(minx+x*5+1,miny+y*5+1,3,3,col);
}
int colval=63;
void Grid::Fill()
{
 int row=0,col=0,flag=1;
 int oldr,oldc;
 oldr=oldc=0;
 char s[4];
 int op;Palette();
 setcolor(MAXCOLORS);
 SBlock(475,15,125,70,0,MAXCOLORS);
 colorText(490,40,"row ",10,1,0,15);
 colorText(490,65,"column ",10,1,0,15);
 sprintf(s,"%d",row+1);
 colorText(555,40,s,4,1,0,15);
 sprintf(s,"%d",col+1);
 colorText(555,65,s,4,1,0,15);
 sprintf(s,"%d",COL+1);
 colorText(XMIN,YMAX+2,s,4,1,0,15);
 sprintf(s,"%d",COL+length+1);
 colorText(XMAX-10,YMAX+2,s,4,1,0,15);
 sprintf(s,"%d",ROW+1);
 colorText(XMAX+5,YMIN,s,3,1,0,15);
 sprintf(s,"%d",ROW+width);
 colorText(XMAX+5,YMAX-15,s,3,1,0,15);
 int color_check[4]={0,0,0,0},cval = 0;//,temp_color=0;
 IniMouse();
 //int count=0;//threadcount=0,count=0;
 buff[row*(length+1)+col] = colval;
 pointer(row-ROW,col-COL);
 op = GetKey();
 while(1)
 {
  setviewport(0,0,getmaxx(),getmaxy(),0);
  pointer(row-ROW,col-COL);
  Click_Fill(col,row,XMIN,YMIN,buff[row*(length+1)+col]);
 loop: switch(op)
  {
   case UP :
                {
                row--;
                break;
                }
   case DOWN :
                {
```

```
             row++;
             break;
             }
case RIGHT :
            {
            col++;
             break;
             }
 case LEFT:
             {
             col--;
             break;
             }
 case HOME:
             {
             row--;
             col--;
             break;
             }

 case END:
           {
           row++;
           col--;
           break;
           }
 case PGUP:
            {
             row--;
             col++;
             break;
             }
 case PGDN:
             {
             row++;
             col++;
             break;
             }
 case ENTER:
              {
              flag++;
              flag%=2;
              break;
              }
 case DELETE:
              {
                int x1,y1;
                x1 = XMIN + col * 5;
                y1 = YMIN + row * 5;
                Block(x1+1,y1+1,3,4,getbkcolor());
                pointer(row-ROW,col-COL);
                op = GetKey();
                buff[row*(length+1)+col]=getbkcolor();
                pointer(row-ROW,col-COL);
                goto loop;
```

```
                  }
     case TAB:
                 {
                  ColorMouse();
                  ShowMouse();
                  Button_Info();
                  while(out_reg.x.bx != 1)
                  {
                     Button_Info();
                  Press=3;
                 if(Press == 3)
                  {
                    if((a>167 && a<423) && (b>13 && b<98))
                         Color_Selection();
                  }
                 }
                  HideMouse();
                  break;
                }
      case ESC:
                 {
                 return;
                 }
}
  colorText(100,40,s,4,1,0,15);
if((row>=width)||(col>=length+1)||(row<0||(col)<0))
{
  row=oldr;
  col=oldc;
  if(oldc==length) col=oldc=length;
  if(oldr==width) row=oldr=width;
  pointer(row-ROW,col-COL);
  sound(300);delay(50);sound(600);delay(50);
  nosound();
  op=GetKey();
  continue;
}

if(row>ROW+width-1)  { ROW+=10;}
if(row<ROW) {ROW-=10;}
if(col<COL+length) { COL+=5;}
if(col<COL) { COL-=5;}
if(ROW>width-width)  { ROW=width-width;}
if(COL>length-length+1) {COL=length-length;}
if(ROW<0) { COL+=10;}
if(COL<0) { COL-=10;}
 if(flag)
  {
  int count=0;
  buff[(row*(length+1))+col]=colval;
  cval = 0;
  for(int i=0;i<4;i++)
      color_check[i] = 0;
  for(i=0;i<width;i++)
   {
```

```
  count = 0;
  if(buff[i*(length+1)+col]!=getbkcolor())
   {
    for(int k=0;k<4;k++)
    {
      if(buff[i*(length+1)+col] == color_check[k])
         break;
       else
         { ++count;
         continue; }
     }
    if(count==4)
    {
      if(cval < 4)
      {
       sprintf(s,"%d",cval+1);
       rectangle(15,88,38,105);
       rectangle(13,86,40,107);
       colorText(XMIN+15,YMIN-17,s,2,1,0,15);
       color_check[cval] = buff[i*(length+1)+col];
       ++cval;
      }
      else
       { ++cval; }
   }
  }
  if(cval>4)
  {
  char mes[20];
  SBlock(175,25,250,70,0,MAXCOLORS);
  strcpy(mes,"YOU HAVE GIVEN > 4 COLORS");
  colorText(200,33,mes,26,1,0,15);
  strcpy(mes,"PLEASE CHOOSE THE SAME COLORS");
  colorText(193,53,mes,29,1,0,15);
  strcpy(mes,"PRESS ENTER...");
  colorText(225,78,mes,20,1,0,15);
  getch();
  Block(165,25,265,81,0);
  Palette();
  for(int i=0;i<4;i++)
    color_check[i] = 0;
  Click_Fill(col,row,XMIN,YMIN,getbkcolor());
  break;
  }
}
}

if(cval > 4)
  {
    pointer(row-ROW,col-COL);
    op = GetKey();
    cval =0;
    goto loop; }
oldr=row;oldc=col;
colorText(490,40,"row ",10,1,0,15);
```

```
 colorText(490,65,"colum ",10,1,0,15);
 sprintf(s,"%d",row+1);
 colorText(555,40,s,4,1,0,15);
 sprintf(s,"%d",col+1);
 colorText(555,65,s,4,1,0,15);
end: pointer(row-ROW,col-COL);
 op=GetKey();
}
}
void Grid::Color_Selection()
{
     pal = getdefaultpalette();
     if(Press==3)
      {
         int coly = (a-167)/60,colx = (b-13)/20;
         colval = pal->colors[(colx*4)+coly];
      }
}
void Grid::Palette()
{
 int i;char msg[10];
 setcolor(WHITE);
 rectangle(XMIN+165,YMIN-95,429,YMIN-10);
 rectangle(XMIN+161,YMIN -97,433,YMIN-8);
             struct palettetype far *pall;
             pall = getdefaultpalette();
             int val = 0;
             for(i=0;i<16;i++)
             {
               if((i%4)==0)
                 val += 19;
               Block(183+(i%4)*60,val,40,15,pall->colors[i]);
               sprintf(msg,"%d",i);
               setcolor(BLUE);
               outtextxy(190+(i%4)*60+15,val+6,msg);
             }

 }
 void Grid::Scale_Img(int Sf_len,int Sf_wth)
 {
   int x = 10,y = 50,tempx,tempy;
   char s[4];
   SBlock(x-3,y-3,(length+1)*Sf_len+4,width*Sf_wth,0,MAXCOLORS);
   colorText(x+145,y+width*Sf_wth+15,"THE        SCALED        IMAGE
)",30,1,0,15);
   sprintf(s,"%d",Sf_len);
   colorText(x+300,y+width*Sf_wth+15,s,2,1,0,15);
   //outtextxy(400,225,s);
   sprintf(s,"%d",Sf_wth);
   colorText(x+350,y+width*Sf_wth+15,s,2,1,0,15);
   for(int i=0;i<width;i++)
   {
     tempx = x;
     for(int j=0;j<=length;j++)
     {
      for(int k=0;k<Sf_len;k++)
      {
        tempy = y;
        for(int l=0;l<Sf_wth;l++)
         {
          putpixel(x,y,buff[j+i*(length+1)]);
          ++y;
         }
        ++x; y = tempy;
      }

     }
     y += Sf_wth;
     x = tempx;
 }
 }
 void Grid::SavePattern()
 {
  char name[45];
  FILE* out;
  SBlock(175,25,250,70,0,MAXCOLORS);
  outtextxy(190,40,"Enter the file name to store");
  InputChar(250,65,&(name[0]),14);
  out=fopen(&(name[0]),"w");
  fwrite((char*)this,1,sizeof(Grid),out);
  fwrite(buff,1,sizeof(char)*length*width,out);
  fclose(out);
  Block(165,25,265,81,0);
  Palette();
  }
 void Grid::GetPattern()
 {
  char name[45];
  FILE* in;
  SBlock(150,150,250,70,0,MAXCOLORS);
  outtextxy(160,160,"Enter the file name to read");
  InputChar(210,188,&(name[0]),14);
  in=fopen(&(name[0]),"r");
  if(in)
  fread((char*)this,1,sizeof(*this),in);
  buff=(char*) calloc(sizeof(char),length*width);
  fread(buff,1,sizeof(char)*length*width,in);
  fclose(in);
  Block(144,150,260,81,0);
  Draw();
  ROW=COL=0;
 }
 void Grid::Cloth_Display()
 {
   ClearViewport(0);
   int colval1,colval2,repeat1,repeat2;
   Palette();
   SBlock(165,200,400,70,0,MAXCOLORS);
   colorText(200,225,"      ENTER WARP SIDE DATA : ",40,1,0,15);
   colorText(220,245,"COLOR VAL:",25,1,0,15);
```

```
    colorText(380,245,"NO. OF REPEAT:",23,1,0,15);
    Input(314,245,colval1,3);
    Input(500,245,repeat1,3);
    cleardevice();
    SBlock(165,200,400,70,0,MAXCOLORS);
    colorText(200,225,"      ENTER WEFT SIDE DATA : ",35,1,0,15);
    colorText(220,245,"COLOR VAL:",20,1,0,15);
    colorText(380,245,"NO. OF REPEAT:",23,1,0,15);
    Input(314,245,colval2,3);
    Input(500,245,repeat2,3);
    ClearViewport(0);
    int st,tempcolval,i,rep;
    for( i=0;i<width;i++)
     {
      if((i%2)==0) st =0;
      else st = 1;
      if(st==1)
      {tempcolval = colval1;colval1=colval2; colval2=tempcolval;}
      for(int j=0;j<=length;j++)
      {
        if(buff[j+i*(length+1)] == getbkcolor())
        {
          if((j%2)==0)
          {
          buff[j+i*(length+1)] = colval1;
          }
          else
           {
            if((j%2)==1)
           buff[j+i*(length+1)] = colval2;
          // Block(XMIN+(j)*5,YMIN+i*5,3,3,(int)buff[j+i*(length+1)]);
           }
        }
    }
    if(st==1)
    {tempcolval = colval1;colval1=colval2; colval2=tempcolval;}
  }
  getch();
  Draw();
  DrawPattern();
  Draw_Img();
}

void Grid::DrawPattern()
{
int i,j;
  for(i=0;i<width;i++)
  for(j=0;j<=length;j++)
     putpixel(8+j,16+i,buff[(j+i*(length+1))]);
}

void Grid::Draw_Img()
{
int i,j;
for(i=0;i<width;i++)
```

```
   for(j=0;j<=length;j++)
      Block(XMIN+1+j*5,YMIN+i*5+1,3,3,(int)buff[j+i*(length-1)]);
 }

 void StartGraph()
 {
  int gd=DETECT,gm,errorno;
  initgraph(&gd,&gm,"c:\\tc\\bgi");
  errorno=graphresult();
  if(errorno!=grOk)
  {
    cout << "Graphics Error:" << grapherrormsg(errorno);
    cout << endl << "press any key to exit";
    cin.get();
    exit(1);
  }
 }

 /*void Block(int left,int top,int len,int ht,int col,int bcol)
 {
   //To set a fill style & color
   if(bcol==17 )
     bcol=col;
   setcolor(bcol);
   setfillstyle(SOLID_FILL,bcol);
   //To set up rectangle coordinates
   int rec[4][2];
   rec[0][0] = rec[3][0] = left;
   rec[0][1] = rec[1][1] = top;
   rec[1][0] = rec[2][0] = left+len;
   rec[2][1] = rec[3][1] = top+ht;
   fillpoly(4,&(rec[0][0]));
 } */

void SBlock(int left,int top,int len,int ht,int col,int bcol=17)
{

  Block(left-5,top+5,len+5,ht+5,col,bcol);
  Block(left,top,len+5,ht+5,col);
  setcolor(bcol);
  rectangle(left+2,top+2,left+len+3,top+ht+3);   //in
  rectangle(left,top,left+len+5,top+ht+5); // out
}
          /*check for void setborder(int)*/
void colorText( int x,int y,char* n,int len,int size,int ForeGround,int Back
{
   settextstyle(DEFAULT_FONT,HORIZ_DIR,size);
   Block(x,y,8*len*size+4,size*8+4,BackGround);
   setcolor(ForeGround);
   outtextxy(x+2,y+2,n);
   setcolor(MAXCOLORS);
}

// To get an integer in graphics mode
void Input(int x,int y,int&n,int len=8)
```

```
{
 const int ENT = 13;
 char s[10] = " ";
 int i=0;
 fflush(stdin);
 colorText(x,y,&(s[0]),len,1,0,15);
 s[i]=getch();
 while( (int)s[i] != ENT)
  {
 i++;
  s[i]='\0';
  colorText(x,y,&(s[0]),len,1,0,15);
  s[i]=toupper(getch());
 if( (int)s[i] == 8)
 {
  if(i!=0) i--;
  s[i]='\0';
  colorText(x,y,&(s[0]),len,1,0,15);
  i--;
 }
}
sscanf(s,"%d",&n);
sprintf(s,"%d",n);
colorText(x,y,&(s[0]),len,1,0,15);
}

void InputChar(int x,int y,char* n,int len=2)
{
  const int ENT = 13;
  int i=0;
  n[0]='\0';
  colorText(x,y,&(n[0]),len,1,0,15);
  n[0]=toupper(getch());
  while( (int)n[i] != ENT)
 {
  n[i+1]='\0';
  colorText(x,y,&(n[0]),len,1,0,15);
  i++;
  n[i]=toupper(getch());
  if((int)n[i] == 8)
 {
  if( i != 0) i--;
  n[i]='\0';
  colorText(x,y,&(n[0]),len,1,0,15);
  i--;
 }
}
n[i]='\0';
colorText(x,y,&(n[0]),len,1,0,15);
}

void SetBorder(int color)
{
 REGS in;
 in.x.ax=0x1001;
 in.h.bl=color;
 int86(0x10,&in,&in);
}

void ClearViewport(int col)
{
 int i=0,j=0;
 setcolor(col);
 while((i!=320)&&(j!=240))
 {
   rectangle(239-i,239-j,400+i,240+j);
   i++;j++;
 }
}
int GetKey()
{
 in_reg.h.ah=8;
 int86(0x21,&in_reg,&out_reg);
 out_reg.h.ah=0;
 if(out_reg.h.al!=0) return(out_reg.x.ax);
 int86(0x21,&in_reg,&out_reg);
 out_reg.h.ah=out_reg.h.al;
 out_reg.h.al=0;
 if(out_reg.x.ax==CTRL_F10)
 {
   closegraph();
   exit(0);
 }
return(out_reg.x.ax);
}

Mou_img::Mou_img(int XMIN,int YMIN,int XMAX,int YMAX,int ROW,int COL) : Grid(
{
  Mou_img::XMIN = XMIN;
  Mou_img::YMIN = YMIN;
  Mou_img::XMAX = XMAX;
  Mou_img::YMAX = YMAX;
  Mou_img::ROW = ROW;
  Mou_img::COL = COL;
}
 void Sound()
 {
   sound(300);delay(50);sound(600);delay(50);
  nosound();
 }
void Mou_img::Image_Display(char *fnl)
{
   char ch,tem_ilen,tem_iwid;
   ifstream fpl;
   fpl.open(fnl,ios::binary);
   if(fpl.fail())
    {
      cerr << "Can't open" << fnl << "file for input";
      exit(1);
    }
    for( int i=0;i<Img_wid;i++)
     for( int j=0;j<3*Img_len;j++)
      {
       if(!fpl.eof())
       {
        fpl.read((char *) &ch,sizeof(char));
        if((ch!=getbkcolor())||(ch!=WHITE))
         putpixel(j-152,i+YMIN,ch/16);
       }
       else
        break;
      }
    fpl.close();
}
int tx,ty,Marker=0,rowl,coll;
 void Mou_img::Fill()
{
 int col=0,row=0;
 char s[4];
 int op;
 colval = 63;
 setcolor(MAXCOLORS);
 Palette();
 Draw();
 SBlock(10,10,125,30,WHITE);
 colorText(20,14,"  QUIT...",11,1,0,15);
 colorText(20,28,"  CLICK OK..!!",13,1,0,15);
 bar3d(60,68,90,105,10,1);
 colorText(60,90,"OK.. ",3,1,0,15);
 SBlock(475,15,125,70,0,MAXCOLORS);
 colorText(490,40,"row ",10,1,0,15);
 colorText(490,65,"column ",10,1,0,15);
 sprintf(s,"%d",col+1);
 colorText(550,65,s,4,1,0,15);
 sprintf(s,"%d",COL+1);
 colorText(XMIN,YMAX+2,s,4,1,0,15);
 sprintf(s,"%d",COL+length+1);
 colorText(XMAX-10,YMAX+2,s,4,1,0,15);
 sprintf(s,"%d",ROW+1);
 colorText(XMAX+5,YMIN,s,3,1,0,15);
 sprintf(s,"%d",ROW+width);
 colorText(XMAX+5,YMAX-15,s,3,1,0,15);
 for(int i=0;i<(length+1)*width;i++)
        buff[i] = getbkcolor();
 ShowMouse();
 Button_Info();
 while(1)
 {
   if((a>=60 && a<=90)&&(b>=60 && b<=90))
   { if(Press==1)
     return;
    }
   if((a>167 && a<423) && (b>13 && b<98))
   {
      Color_Selection();
```

```
        goto nos;
    }
    col = (a-XMIN)/5;row=(b-YMIN)/5;
    if(((out_reg.x.bx==1)||(out_reg.x.bx==2)||(out_reg.x.bx==4))
                              &&((a>=XMIN && a<XMAX) && (b>=YMIN && b<YMAX)))
    {
         if(out_reg.x.bx==4)
          {
            FillPattern(col,row);
            goto nos;
          }
          FillPattern(col,row);
          buff[row*(length+1)+col]= colval;
    }
                 // left - 1 (Fill),right - 2 (Defill)
  if(row<0) {row = 0;Sound();}
  if(col<0) {col = 0;Sound();}
  if(row > width) {row = width;Sound();}
  if(col > length+1) {col = length;Sound();}
  sprintf(s,"%d",row+1);
  colorText(550,40,s,4,1,0,15);
  sprintf(s,"%d",col+1);
  colorText(550,65,s,4,1,0,15);
  if((row>=width)||(col>=length+1)||(row<0)||(col<0))
 {
   //sound(300);delay(50);sound(600);delay(50);
   //nosound();
   goto nos;
 }
  if((out_reg.x.bx==1)||(out_reg.x.bx==2))
   buff[row*(length+1)+col]= colval;
  if(row>ROW+width)  { ROW+=10;}
  if(row<ROW) {ROW-=10;}
  if(col<COL+length) { COL+=5;}
  if(col<COL) { COL-=5;}
  if(ROW>width-width)  { ROW=width-width;}
  if(COL>length-length) {COL=length-length;}
  if(ROW<0) { COL+=10;}
  if(COL<0) { COL-=10;}
  if(ROW < 0) ROW = 0;
  if(COL < 0) COL = 0;
  nos: col=row=0;Button_Info();
 }
 }
void Mou_img::Threadcount()
{
  Grid::Threadcount();
}
 void Mou_img::Nothread_Display()
 {
   Grid::Nothread_Display();
 }
 void Mou_img::Scale_Img(int Sf_len,int Sf_wth)
 {
  Grid::Scale_Img(Sf_len,Sf_wth);
  }
  void Mou_img::Nothread()
  {
  Grid::Nothread();
  }
  void Mou_img::Split_Img()
  {
  Grid::Split_Img();
  }
  void Mou_img::Color_Selection()
  {
  Grid::Color_Selection();
  }
  void Mou_img::SavePattern()
  {
  Grid::SavePattern();
  }
  void Mou_img::GetPattern()
  {
  Grid::GetPattern();
  }
  void Mou_img::Draw_Img()
  {
  Grid::Draw_Img();
  }
  void Mou_img::Cloth_Display()
  {
  Grid::Cloth_Display();
  }
 void Mou_img::Set(int x,int y,char *Warp)
 {
    char fnl[15];
    Mou_img::length=x;Mou_img:;width=y;
    color=Warp;ROW=COL=0,colorno=1;
    XMIN=2;XMAX=XMIN+(length+1)*5;YMIN=108;YMAX=YMIN+width*5;
    SBlock(165,200,400,70,0,MAXCOLORS);
    colorText(200,225,"enter the image file name : ",40,1,0,15);
    InputChar(325,245,fnl,15);
    Block(163,198,400,70,0,MAXCOLORS);
    SBlock(165,200,400,70,0,MAXCOLORS);
    colorText(200,225,"     IMAGE SIZE : ",25,1,0,15);
    colorText(200,245,"LENGTH : ",10,1,0,15);
    Input(325,245,Img_len,8);
    colorText(200,245,"WIDTH  : ",10,1,0,15);
    Input(325,245,Img_wid,8);
    Block(163,198,400,70,0,MAXCOLORS);
    ClearViewport(0);
    buff=(char *) calloc(sizeof(char),(x+1)*y);
    for(x=0;x<(length+1)*width;x++)
         buff[x] = getbkcolor();
    setpalette(0,7);
    unic = (char *) calloc(sizeof(char),16);
    unioccur = (int *) calloc(sizeof(int),16);
    Image_Display(fnl);
}
```

```
void Mou_img::FillPattern(int x,int y)
{
  if(Press==1)
  {
    HideMouse();
    Click_Fill(x,y,XMIN,YMIN,colval);
    ShowMouse();
  }
  if(Press==2)
  {
     HideMouse();
    Click_Fill(x,y,XMIN,YMIN+1,getbkcolor());
    line(XMIN+x*5+1,YMIN+y*5+1,XMIN+x*5+4,YMIN+y*5+1);
    ShowMouse();
  }
  if(Press==4)
  {
   ++Marker;
   HideMouse();
   Click_Fill(x,y,XMIN,YMIN,colval);
   if(Marker==1)
     { tx = x;ty = y; }
   if(Marker==2)
   {
     Press=0;
     Marker=0;
       coll = (a-XMIN)/5;rowl=(b-YMIN)/5;
       int tt=0;
       if(ty>rowl)
         { tt = ty;ty = rowl;rowl = tt; }
     for(int i=ty;i<=rowl;i++)
     {  for(int j=tx;j<=coll;j++)
       {  buff[i*(length+1)+j] = (char)colval;//getbkcolor();
          Click_Fill(j,i,XMIN,YMIN,buff[j+i*(length+1)]);
          line(XMIN+j*5+1,YMIN+i*5+4,XMIN+j*5+4,YMIN+i*5+4);
       }
     }       //getbkcolor()); }}
     tx=0;ty=0;coll=rowl=0;
       }
     }
 ShowMouse();
 }

void IniMouse()
{
 in_reg.x.ax=0x0000;
 in_reg.x.bx=0x0003;
 int86(0x33,&in_reg,&out_reg);
}

/* TO CONFINE THE MOUSE TO THE PALETTE */
void ColorMouse()
{
 //      For X - Coordinates
```

```
  in_reg.x.ax = 7;
  in_reg.x.cx = 167;
  in_reg.x.dx = 423;
  int86(0x33,&in_reg,&out_reg);
  //       For Y - Coordinates
  in_reg.x.ax = 8;
  in_reg.x.cx = 13;
  in_reg.x.dx = 98;
  int86(0x33,&in_reg,&out_reg);
}
void ShowMouse()
{
  in_reg.x.ax=0x0001;
  int86(0x33,&in_reg,&out_reg);
}
void HideMouse()
{
  in_reg.x.ax=0x0002;
  int86(0x33,&in_reg,&out_reg);
}


void Button_Info()
{
 in_reg.x.ax = 0x0003;
 int86(0x33,&in_reg,&out_reg);
 Press = out_reg.x.bx;
 a = out_reg.x.cx;
 b = out_reg.x.dx;
}
void Copy(ffblk *out,ffblk *in)
{
 strcpy(out->ff_reserved,in->ff_reserved);
 strcpy(out->ff_name,in->ff_name);
 out->ff_attrib=in->ff_attrib;
 out->ff_ftime=in->ff_ftime;
 out->ff_fdate=in->ff_fdate;
 out->ff_fsize=in->ff_fsize;
}

void TDir(char* type,char* title,char* name)
{
 ffblk fname,first;
 int count=0;
 int found;
 int i=0;

 found=findfirst(type,&fname,0);
 if(!found)
  {
   Copy(&first,&fname);
   count++;
  }
 setcolor(WHITE);
 SBlock(180,180,270,70,LIGHTGREEN,WHITE);
 colorText(200,176,&(title[0]),19,1,0,15);
 colorText(185,200,"Selected file",14,1,LIGHTGREEN,BLUE);
 colorText(320,200,&(fname.ff_name[0]),14,1,RED,WHITE);
 while(1)
 switch(GetKey())
 {
  case DOWN:found=findnext(&fname);
            if(found)
            {
             sound(250);
             delay(100);
             sound(500);
             delay(100);
             nosound();
             break;
            }
            count++;
            colorText(320,200,&(fname.ff_name[0]),14,1,RED,WHITE);
            break;
  case UP  :
            if(count>0)
            {
            found=findfirst(type,&fname,0);
            count--;
            for(i=0;i<count;i++) findnext(&fname);
            }
            else
            {
             sound(250);
             delay(100);
             sound(500);
             delay(100);
             nosound();
             break;
            }
            colorText(320,200,&(fname.ff_name[0]),14,1,RED,WHITE);
            break;
  case ENTER:
            strcpy(name,&(fname.ff_name[0]));
            Block(170,180,300,90,0,0);
            return;
  default:  continue;
 }

}

void Dir(char* type,char* title)
{
 ffblk fname;
 int found;
 int i=0;
 found=findfirst(type,&fname,0);
 setcolor(WHITE);
 rectangle(99,199,401,321);
 rectangle(97,197,403,323);
 SBlock(100,200,300,120,LIGHTGREEN,WHITE);
 colorText(175,192,&(title[0]),19,1,0,15);
 while(!found)
 {
  //cout<<fname.ff_name<<endl;
  colorText(101+(int)(i/8)*150,205+(i%8)*12,&(fname.ff_name[0]),
      14,1,BLUE,LIGHTGREEN);
  i++;
  found=findnext(&fname);
  if(i==16)
  {
   colorText(110,305,"Press a key to continue",30,1,LIGHTGREEN,BLUE);
   getch();
   setcolor(WHITE);
   rectangle(99,199,401,321);
   rectangle(97,197,403,323);
   Block(100,200,300,120,LIGHTGREEN,WHITE);
   colorText(175,192,&(title[0]),8,1,0,15);
   i=0;
  }
 }
   colorText(110,309,"Press a key to continue",25,1,0,15);
   getch();
   Block(91,190,403,323,0);
}
void IconDraw(int row,int colum,char *s,int size=1,
           int text=WHITE,int back=LIGHTGRAY,int border=DARKGRAY)
{
  int i,j,k,n,p,r,c;
  int a[12];
  //int cn=0;
  //i=0;
  n=0;
  while(s[n]!='\0') n++;
  Block(row-2,colum-2,n*size*8+4,size*8+4,border,border);
  Block(row,colum,n*size*8,size*8,back,back);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,size);
  setcolor(text);
  outtextxy(row+7,colum-5,s);
 }
/*void IconDraw(int row,int colum,char *s,int size=1,int font=1,
           int text=WHITE,int back=LIGHTGRAY,int border=DARKGRAY)
{
  int i,j,k,n,p,r,c;
  int a[12];
  int cn=0;
  i=0;
  n=0;
  while(s[n]!='\0') n++;
  Block(row-2,colum-2,n*size*8+4,size*8+4,border,border);
  Block(row,colum,n*size*8,size*8,back,back);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,size);
  setcolor(text);
  outtextxy(row+7,colum-5,s);
 }*/
```

```
int menu::Indv_check(int disl,int ind)
{    int j;
 if(disl==1)
  for(j=1;j<ind;j++)
if((a>D_Menu[j][0] && a<D_Menu[j][0]+75)&&(b>D_Menu[j][1]&&b<D_Menu[j][1]+30
     return(j);
 if(disl==2)
  for(j=1;j<ind;j++)
  if((a>Demo[j][0] && a<Demo[j][0]+75)&& (b>Demo[j][1] && b<Demo[j][1]+30))

     return(j);
 if(disl==3)
  for(j=1;j<ind;j++)
  if((a>Help[j][0] && a<Help[j][0]+75)&&(b>Help[j][1] && b<Help[j][1]+30))

     return(j);
 if(disl==11)
  for(j=1;j<ind;j++)
  if((a>C_Keys[j][0] && a<C_Keys[j][0]+75)&&
                                  (b>C_Keys[j][1] && b<C_Keys[j][1]+30))

     return(j);
 if(disl==12)
  for(j=1;j<ind;j++)
  if((a>Mouse[j][0] && a<Mouse[j][0]+75)&&(b>Mouse[j][1] && b<Mouse[j][1]+30

     return(j);
 if(disl==13)
  for(j=1;j<ind;j++)
  if((a>View[j][0] && a<View[j][0]+75)&&(b>View[j][1] && b<View[j][1]+30))

     return(j);
 if(disl==14)
  for(j=1;j<ind;j++)
  if((a>Sub_Help[j][0] && a<Sub_Help[j][0]+75)&&
                                  (b>Sub_Help[j][1] && b<Sub_Help[j][1]+30))
     return(j);
 return(100);
}
void menu::Set_Val()
{
  menu::D_Menu[0][0] = 125;menu::D_Menu[0][1] = 34;
  menu::D_Menu[1][0] = 125;menu::D_Menu[1][1] = 65;menu::D_Menu[2][0] = 125;
  menu::D_Menu[2][1] = 96;menu::D_Menu[3][0] = 125;menu::D_Menu[3][1] = 127;
  menu::Demo[0][1] = 286;Demo[0][1] = 34;
  menu::Demo[1][0] = 286; menu::Demo[1][1] = 65;menu::Demo[2][0] = 286;
  menu::Demo[2][1] = 96;menu::Help[0][0] = 442;menu::Help[0][1] = 34;
  menu::Help[1][0] = 442;menu::Help[1][1] = 65;
  menu::Help[2][0] = 442;menu::Help[2][1] = 96; menu::Help[3][0] = 442;
  menu::Help[3][1] = 127;menu::Help[4][0] = 442;menu::Help[4][1] = 158;
  menu::Help[5][0] = 442;menu::Help[5][1] = 189;
  menu::C_Keys[0][0] =110;menu::C_Keys[0][1] = 34;
  menu::C_Keys[1][0] = 110;menu::C_Keys[1][1] = 65;menu::C_Keys[2][0] =110;
  menu::C_Keys[2][1] = 96;menu::C_Keys[3][0] = 110;menu::C_Keys[3][1] = 127;
  menu::C_Keys[4][0] = 110;menu::C_Keys[4][1] = 158;menu::C_Keys[5][0] = 110;
```

```
   menu::Mouse[0][0] = 261;menu::C_Keys[5][1] = 189;
   menu::Mouse[0][1] = 34;menu::Mouse[1][0] = 261;menu::Mouse[1][1] = 65;
   menu::Mouse[2][0] = 261;menu::Mouse[2][1] = 96;menu::Mouse[3][0] = 261;
   menu::Mouse[3][1] = 127;menu::Mouse[4][0] = 261;menu::Mouse[4][1] = 158;
   menu::Mouse[5][0] = 261;menu::Mouse[5][1]  = 189;
   menu::View[0][0] = 412;menu::View[0][1] = 34;menu::View[1][0] = 412;
   menu::View[1][1] = 65;menu::View[2][0] = 412;menu::View[2][1] = 96;
   menu::Sub_Help[0][0] = 412;menu::Sub_Help[0][1] = 34;
   menu::Sub_Help[1][0] = 412;menu::Sub_Help[1][1] = 65;
   menu::Sub_Help[2][0] = 412;menu::Sub_Help[2][1] = 96;
 }
 void menu::Display(int dis,int index)
 {
   int j;
   setfillstyle(0,13);
   settextstyle(1,0,1);
   setcolor(9);
   HideMouse();
  if(dis==1)
  {
   for(j=1;j<index;j++)
     Block(D_Menu[j][0],D_Menu[j][1]+5,75,30,3);
   setcolor(9);
   outtextxy(D_Menu[1][0]+2,D_Menu[1][1]+10,"C-KEYS");
   outtextxy(D_Menu[2][0]+2,D_Menu[2][1]+10,"MOUSE");
   outtextxy(D_Menu[3][0]+2,D_Menu[3][1]+10,"EXIT");
  }
  if(dis==2)
  {
   for(j=1;j<index;j++)
     Block(Demo[j][0],Demo[j][1],75,30,3);
   setcolor(9);
   outtextxy(Demo[1][0],Demo[1][1],"DEMO");
   outtextxy(Demo[2][0],Demo[2][1],"COLOR256");
}

 if(dis==3)
 {
  for(j=1;j<index;j++)
    Block(Help[j][0],Help[j][1],75,30,3);
  setcolor(9);
    outtextxy(Help[1][0],Help[1][1],"WEAVING");
    outtextxy(Help[2][0],Help[2][1],"POWERLOOM");
    outtextxy(Help[3][0],Help[3][1],"LICENCE");
    outtextxy(Help[4][0],Help[4][1],"HARDWARE");
    outtextxy(Help[5][0],Help[5][1],"SOFTWARE");
 }

if(dis==11)
 {
 for(j=1;j<index;j++)
   Block(C_Keys[j][0],C_Keys[j][1],75,30,3);
   setcolor(9);
 outtextxy(C_Keys[1][0],C_Keys[1][1],"CREATE");
 outtextxy(C_Keys[2][0],C_Keys[2][1],"SAVE");
```

```
   outtextxy(C_Keys[3][0],C_Keys[3][1],"LOAD");
   outtextxy(C_Keys[4][0],C_Keys[4][1],"DIR");
   outtextxy(C_Keys[5][0],C_Keys[5][1],"MAIN MENU");
  }

  if(dis==12)
  {
   for(j=1;j<index;j++)
     Block(Mouse[j][0],Mouse[j][1],75,30,3);
     setcolor(9);
   outtextxy(Mouse[1][0],Mouse[1][1],"CREATE");
   outtextxy(Mouse[2][0],Mouse[2][1],"SAVE");
   outtextxy(Mouse[3][0],Mouse[3][1],"LOAD");
   outtextxy(Mouse[4][0],Mouse[4][1],"DIR");
   outtextxy(Mouse[5][0],Mouse[5][1],"RETURN");
  }

  if(dis==13)
  {
   for(j=1;j<index;j++)
     Block(View[j][0],View[j][1],75,30,3);
     setcolor(9);
   outtextxy(View[1][0],View[1][1],"ZOOM IN");
   outtextxy(View[2][0],View[2][1],"ZOOM OUT");
  }

  if(dis==14)
  {
   for(j=1;j<index;j++)
     Block(Sub_Help[j][0],Sub_Help[j][1],75,30,3);
     setcolor(9);
   outtextxy(Sub_Help[1][0],Sub_Help[1][1],"C-KEYS");
   outtextxy(Sub_Help[2][0],Sub_Help[2][1],"MOUSE");
}
setcolor(15);
ShowMouse();
}
int cr, Pageno = 0;
int menu::Mouse_check()
{
int i;
 if(Pageno==0)
 {
  for( i=0;i<4;i++)
   if((a>D_Menu[i][0] && a<D_Menu[i][0]+75)
                                  &&(b>D_Menu[i][1]&&b<D_Menu[i][1]+30))
      return(1);
  for(i=0;i<3;i++)
   if((a>Demo[i][0] && a<Demo[i][0]+75)&& (b>Demo[i][1] && b<Demo[i][1]+30))
      return(2);
  for(i=0;i<6;i++)
    if((a>Help[i][0] && a<Help[i][0]+75)&&(b>Help[i][1] && b<Help[i][1]+30))
      return(3);
 }
 if(Pageno==1)
 {
 for(i=0;i<6;i++)
 if((a>C_Keys[i][0]&&a<C_Keys[i][0]+75)&&(b>C_Keys[i][1] &&b<C_Keys[i][1]+3
      return(11);
 for( i=0;i<6;i++)
 if((a>Mouse[i][0] && a<Mouse[i][0]+75)&&(b>Mouse[i][1] && b<Mouse[i][1]+30
      return(12);
      for(i=0;i<4;i++)
 if((a>View[i][0] && a<View[i][0]+75)&&(b>View[i][1] && b<View[i][1]+30))
      return(13);
      for(i=0;i<3;i++)
 if((a>Sub_Help[i][0] && a<Sub_Help[i][0]+75)&&
                                   (b>Sub_Help[i][1] && b<Sub_Help[i][1]+30))
      return(14);
}
return(0);
}


void menu::Disp_Page1()
{
HideMouse();
 ClearViewport(0);
 setcolor(WHITE);
  rectangle(0,2,getmaxx(),getmaxy());
 rectangle(2,4,getmaxx()-2,getmaxy()-2);
 rectangle(3,5,getmaxx()-3,27);
 Block(5,6,getmaxx()-10,20,getbkcolor());
 setcolor(WHITE);
 settextstyle(1,0,2);
 outtextxy(170,6," C A T E X - D E S I G N ");
 rectangle(3,32,getmaxx()-3,56);
 //rectangle(2,24,getmaxx()-2,38);
 Block(4,33,getmaxx()-10,22,getbkcolor());
 setcolor(WHITE);
 outtextxy(125,34,"DOUBLE");
 outtextxy(285,34,"DEMO");
 outtextxy(442,34,"HELP");
 ShowMouse();
}

void menu::Disp_Page2()
{
 HideMouse();
 ClearViewport(0);
 setcolor(WHITE);
  rectangle(0,2,getmaxx(),getmaxy());
 rectangle(2,4,getmaxx()-2,getmaxy()-2);
 rectangle(3,5,getmaxx()-3,27);
 Block(4,6,getmaxx()-10,20,getbkcolor());
 setcolor(WHITE);
 settextstyle(1,0,2);
 outtextxy(170,6," C A T E X - D E S I G N ");
 rectangle(3,32,getmaxx()-3,56);
 Block(4,33,getmaxx()-10,22,getbkcolor());
 setcolor(WHITE);
 outtextxy(110,34,"C-KEYS");
 outtextxy(261,34,"MOUSE");
 outtextxy(412,34,"VIEW");
  outtextxy(563,34,"HELP");
  ShowMouse();
}

void main()
{
 int option;
 char *r;
 system("tittt");
 getch();
 StartGraph();
 Help help;
 menu mmenu;
 int Pick,Read;
 rectangle(0,2,getmaxx(),getmaxy());
 rectangle(2,4,getmaxx()-2,getmaxy()-2);
 Block(3,45,getmaxx()-7,getmaxy()-48,LIGHTGRAY);
 int indvi=0;
 Button_Info();
 setcolor(MAXCOLORS);
 mmenu.Disp_Page1();
   IniMouse();
   ShowMouse();
   r = (char *)calloc(sizeof(char),16);
   Grid Ck_img(2,108,Pick*5,Read*5,0,0);
   Mou_img Mouse(2,108,Pick*5,Read*5,0,0);
   mmenu.Set_Val();
  while(1)
  {
//     getch();
     Button_Info();
     if(b<27)
     {
        HideMouse();
        Block(2,58,getmaxx()-8,getmaxy()-60,LIGHTGRAY);
        ShowMouse();
     }
     if((Pageno==0)||(Pageno==1))
     option = mmenu.Mouse_check();
    switch(option)
    {
     case 1:
            {
             mmenu.Display(option,4);
       //            getch();
             Button_Info();
             if(Press==1)
              indvi = mmenu.Indv_check(option,3);
             if((Press==1)&&((indvi==0)||(indvi==1)))
               ++Pageno;
             if(Pageno==1)
        { mmenu.Disp_Page2(); }
        if((Press==1)&&(indvi==2))
         { closegraph();exit(1); }
        break;
        }
case 2:
        {
         mmenu.Display(option,3);
         Button_Info();
         if(Press==1)
         indvi = mmenu.Indv_check(option,3);
         if((Press==1)&&(indvi==1))
           Color256();
         break;
        }
case 3:
        {
         char name[20];
         mmenu.Display(option,6);
         Button_Info();
         if(Press==1)
        indvi = mmenu.Indv_check(option,3);
         if((Press==1)&&(indvi==0))
         {
            strcpy(name,"weav.dat");
            help.New(name, "  WEAVING");
            help.TraceHelp();
         }
        if((Press==1)&&(indvi==1))
         {
            strcpy(name,"powl.dat");
            help.New(name, "  POWER LOOM");
            help.TraceHelp();
         }
         if((Press==1)&&(indvi==1))
         {
            strcpy(name,"lic.endat");
            help.New(name, "  LICENCE");
            help.TraceHelp();
         }
         break;
       }
case 11:
         {
           mmenu.Display(option,6);
           delay(3000);
           //getch();
           Button_Info();
           if(Press==1)
          indvi = mmenu.Indv_check(option,3);
           if(Press==1)
           {
    switch(indvi)
    {
      case 0:
```

```
                    {
                       HideMouse();
                       SBlock(165,200,400,70,0,MAXCOLORS);
                       colorText(200,225,"READ VALUE...(from Table values) :
                       Input(325,245,Read,5);
                       cleardevice();
                       SBlock(165,200,400,70,0,MAXCOLORS);
                       colorText(200,225,"PICK VALUE...(from Table values) :
                       Input(325,245,Pick,5);
                       Ck_img.Set(Pick-1,Read,r);
                       Ck_img.Draw();
                       Ck_img.Shade();
                       Ck_img.Fill();
                    }
            case 1: {Ck_img.SavePattern();}
            case 2: Ck_img.GetPattern();
            case 3:
            case 4: {mmenu.Disp_Page1();Pageno=0;}
          }
        }
       //        ClearViewport(0);
        ShowMouse();
        break;
       }
   case 12:
      {
        mmenu.Display(option,6);
        delay(3000);
        //getch();
        Button_Info();
        if(Press==1)
        indvi = mmenu.Indv_check(option,3);
        if(Press==1)
        {
          switch(indvi)
          {
            case 0:
                 {
         ClearViewport(0);
         SBlock(165,200,400,70,0,MAXCOLORS);
         colorText(200,225,"READ VALUE...(from Table values) : ",40,1,0,15);
         Input(325,245,Read,5);
         cleardevice();
         SBlock(165,200,400,70,0,MAXCOLORS);
         colorText(200,225,"PICK VALUE...(from Table values) : ",40,1,0,15);
         Input(325,245,Pick,5);
         Mouse.Set(Pick-1,Read,r);
         Mouse.Fill();
         Block(165,25,265,81,0);
         Mouse.Palette();
}
   case 1: Mouse.SavePattern();
   case 2: Mouse.GetPattern();
   case 3:
   case 4: mmenu.Disp_Page2();
```

```
  }
}
       //ClearViewport(0);
               break;
              }
      case 13:
             {
               mmenu.Display(option,3);
               Button_Info();
               if(Press==1)
            indvi = mmenu.Indv_check(option,3);
               int S_len,S_wid;
               if(Press==1)
               {
                 switch(indvi)
                 {
                   case 0: Ck_img.Draw_Img();
                   case 1: Ck_img.Scale_Img(S_len,S_wid);
                 }
               }
               break;
             }
     case 14:
             {
               char name[20];
               mmenu.Display(option,3);
               Button_Info();
               if(Press==1)
               indvi = mmenu.Indv_check(option,3);
               if((Press==1)&&(indvi==0))
               {
                strcpy(name,"c-keys.dat");
                help.New(name, "  CTRL-KEYS");
                help.TraceHelp();
             }
            if((Press==1)&&(indvi==1))
             {
                strcpy(name,"mouse.dat");
                help.New(name, "  MOUSE");
                help.TraceHelp();
             }
             break;
              }
     /*case 0:
            {
            HideMouse();
            Block(4,56,getmaxx()-8,getmaxy()-60,LIGHTGRAY);
            setcolor(WHITE);
            ShowMouse();
            continue;
            }   */
    default:     continue;
  }
  HideMouse();
  Block(2,58,getmaxx()-8,getmaxy()-60,getbkcolor());
```

```
   setcolor(WHITE);
//   delay(6000);
   ShowMouse();
   option=0;
}
}




/*void main()
{
 Fabric  fab;
 Color256();
 IniMouse();
 fab.SetPalette();
// fab.ChangePalette();
 fab.RestorePalette();
 fab.Palette();
 clearviewport();
// Info();
 //getch();
 p();
 getch();
 clearviewport();

} */


/*  Write Function for Setting Standard Background Colour instead of 16

void main()
{
 char *r;
 int key,option,x1,y1,x2,y2;
 int gd=DETECT,gm,errorno;
 printf("\n=============OPTION============");
 printf("\n  1  ----->  USING CONTROL KEYS  ");
 printf("\n  2  ----->  USING MOUSE         ");
 printf("\n  3  ----->  EXIT                ");
 printf("\n=================================");
 printf("\nENTER YOUR OPTION  :  ");
 cin >> option;
  StartGraph();
  int Read,Pick;
   SBlock(165,200,400,70,0,MAXCOLORS);
   colorText(200,225,"READ VALUE...(from Table values) : ",40,1,0,15);
   Input(325,245,Read,5);
   cleardevice();
   SBlock(165,200,400,70,0,MAXCOLORS);
   colorText(200,225,"PICK VALUE...(from Table values) : ",40,1,0,15);
   Input(325,245,Pick,5);
   cleardevice();
```

```
switch(option)
{
  case 1:
         {
           Grid Ck_img(2,108,Pick*5,Read*5,0,0);
           Ck_img.Set(Pick-1,Read,r);
           Ck_img.Draw();
           Ck_img.Shade();
           Ck_img.Fill();
 /*  Ck_img.GetPattern();
   Ck_img.Draw_Img();
   Ck_img.DrawPattern();
   Ck_img.Threadcount();
   Ck_img.SavePattern();
   getch();
   int S_len,S_wth;
   ClearViewport(0);
   SBlock(165,200,400,70,0,MAXCOLORS);
   colorText(200,225,"ENTER THE SCALING FACTOR OF LENGTH : ",40,1,0,15);
   Input(325,245,S_len,5);
   cleardevice();
   SBlock(165,200,400,70,0,MAXCOLORS);
   colorText(200,225,"ENTER THE SCALING FACTOR OF WIDTH : ",40,1,0,15);
   Input(325,245,S_wth,5);
   ClearViewport(0);
   Ck_img.Scale_Img(S_len,S_wth);
   Ck_img.Nothread();
   Ck_img.Nothread_Display();
   getch();
   break;
 }
 case 2:
        {
         Mou_img Mouse_img(2,108,Pick*5,Read*5,0,0);      char save[2];
         Mouse_img.Set(Pick-1,Read,r);
         Mouse_img.Fill();
         Block(165,25,265,81,0);
         Mouse_img.Palette();
         Mouse_img.Threadcount();
         Mouse_img.SavePattern();
         HideMouse();
          getch();
          int S_len,S_wth;
          ClearViewport(0);
          SBlock(165,200,400,70,0,MAXCOLORS);
  colorText(200,225,"ENTER THE SCALING FACTOR OF LENGTH : ",40,1,0,15);
          Input(325,245,S_len,5);
          cleardevice();
          SBlock(165,200,400,70,0,MAXCOLORS);
  colorText(200,225,"ENTER THE SCALING FACTOR OF WIDTH : ",40,1,0,15);
          Input(325,245,S_wth,5);
          ClearViewport(0);
        Mouse_img.Scale_Img(S_len,S_wth);
        Mouse_img.GetPattern();
        Mouse_img.Draw_Img();
```

```
                Mouse_img.Nothread();
                Mouse_img.Nothread_Display();
                 getch();
               break;
            }
     }
 closegraph();
 exit(1);
 }*/
```

```
#include<stdio.h>
# include<iostream.h>
# include<alloc.h>
# include<string.h>
#include<dos.h>
#include <graphics.h>
#include <conio.h>
int l=0;
void Block(int left,int top,int len,int ht,int col,int bcol=17);
class PROJ_RAIN
{
  private:
            union REGS regs;
            char *G_Name;
            char *G_by,*D_by;
            char *P_Name;
            char *P1_Name,*P2_Name,*P3_Name;
            char *a;
  public:
            PROJ_RAIN();
            void Calldisp();
            void Dis(char *temp,int start,int ypos);
            void Cursoroff();
            void Cursoron();
            void Thank_you();
};
PROJ_RAIN::PROJ_RAIN()
{
  G_Name = (char *)calloc(sizeof(char),25);
  strcpy(G_Name,"Mr. M.GOPALAN B.E.,M.S.");
  G_by = (char *)calloc(sizeof(char),10);
  strcpy(G_by,"GUIDED BY");
  D_by = (char *)calloc(sizeof(char),10);
  strcpy(D_by,"DONE BY");
  P_Name = (char *)calloc(sizeof(char),25);
  strcpy(P_Name,"CATEX DESIGN");
  P1_Name = (char *)calloc(sizeof(char),25);
  strcpy(P1_Name,"S. VIJAYARAGAVAN.");
  P2_Name = (char *)calloc(sizeof(char),25);
  strcpy(P2_Name,"P. SUDHA.");
  P3_Name = (char *)calloc(sizeof(char),25);
  strcpy(P3_Name,"N. SASIKALA.");
  a = (char *)calloc(sizeof(char),25);
}
void Block(int left,int top,int len,int ht,int col,int bcol)
{
  //To set a fill style & color
  if(bcol==17 )
    bcol=col;
  setcolor(bcol);
  setfillstyle(SOLID_FILL,bcol);
  //To set up rectangle coordinates
  int rec[4][2];
  rec[0][0] = rec[3][0] = left;
  rec[0][1] = rec[1][1] = top;
  rec[1][0] = rec[2][0] = left+len;
  rec[2][1] = rec[3][1] = top+ht;
  fillpoly(4,&(rec[0][0]));
}
void PROJ_RAIN::Cursoroff()
{
regs.h.ah = 1;
regs.h.ch = 8;
regs.h.cl = 0;
int86(0x10,&regs,&regs);
}
void PROJ_RAIN::Thank_you()
  {
    int a,b,i=1;
    do{
    i++;
    outtextxy(100,100,"THANK");
    outtextxy(100,250,"    YOU");
    sleep(1);
    cleardevice();
    }while(i < 5);
  }
void PROJ_RAIN::Calldisp()
{
  Dis(P3_Name,15,19);
  Dis(P2_Name,15,17);
  Dis(P1_Name,15,15);
  Dis(D_by,5,13);
  Dis(G_Name,12,11);
  Dis(G_by,5,8);
}
void PROJ_RAIN::Dis(char *a,int startx,int ypos)
{
    int c,j,i;
    char word;
    int xloc,yloc,count;
    j=0;int len = strlen(a)/2;
    int rowval = startx;
    int  t_wd,t_ht,k;
    t_wd = textwidth("W");
    t_ht = textheight("W");
    char st[1],str[1];
    strcpy(str," ");
    for(j=0;j<strlen(a)/2;j++)
    {
     for(int i=0;i<ypos;i++)
     {
      st[0] = *(a+j);
      st[1] = '\0';
      outtextxy((startx+j)*t_wd,i*t_ht,st);
      delay(10);
      st[0] = *(a+(strlen(a)-j-1));
      st[1] = '\0';
      k = strlen(a)-j-1;
      outtextxy((startx+k)*t_wd,i*t_ht,st);
      delay(10);
      Block(0,0,getmaxx(),(ypos-1)*t_ht+3,0);
      setcolor(WHITE);
     }

    }
    if(strlen(a)/2!=0)
      {
      st[0] = *(a+(strlen(a)/2));
     for(int i=0;i<ypos;i++)
     {
      st[1] = '\0';
      outtextxy((startx+j)*t_wd,i*t_ht,st);
      Block(0,0,getmaxx(),(ypos-1)*t_ht+3,0);
      setcolor(WHITE);
      delay(10);
      }
    }
}

void PROJ_RAIN::Cursoron()
{

regs.h.ah = 1;
regs.h.ch = 7;
regs.h.cl = 8;
int86(0x10,&regs,&regs);
}

void main()
{
  int i,j,k,n,gd=DETECT,gm;
  PROJ_RAIN CRAIN;
  initgraph(&gd,&gm," ");
  clearviewport();
  settextstyle(1,0,3);
  CRAIN.Calldisp();
  getch();
  //CRAIN.Thank_you();
  closegraph();
 }
```

```
void movvl(int c[4][80],int k);
void movv2(int d[4][80],int k);
void main()
{
int a[4][80];
int b[4][80];
int i;
clrscr();
for(i=0;i<=7;i++)
{
a[4][i] = 255;
}
a[4][8]= 223;  a[4][9] = 255;
a[4][10] = 223;a[4][11] = 255;
a[4][12] = 223;a[4][13] = 223;
for(i=14;i<=16;i++)
{
a[4][i] = 255;
}
a[4][17] = 223;
for(i=18;i<=26;i++)
{
a[4][i] = 255;
}
a[4][27] = 223;
for(i=28;i<=33;i++)
{
a[4][i] = 255;
}
a[4][34] = 223;
for(i=35;i<=42;i++)
{
a[4][i] = 255;
}
a[4][43] = 223;a[4][44] = 255;
a[4][45] = 223;a[4][46] = 255;
a[4][47] = 223;a[4][48] = 255;
a[4][49] = 223;a[4][50] = 222;
for(i=51;i<=53;i++)
{
a[4][i] = 255;
}
a[4][54] = 223;a[4][55] = 223;
for(i=56;i<=61;i++)
{
a[4][i] = 255;
}
a[4][62] = 223;a[4][63] = 223;
for(i=64;i<=80;i++)
{
a[4][i] = 255;
}
movvl(a,4);
//------------------
for(i=0; i<6; i++)
```

```
{
b[4][i] = 255;
}
b[4][6] =220;
b[4][7] =b[4][9]=255;
b[4][8] =b[4][10]=220;
for( i=11; i<17; i++)
{
b[4][i] = 255;
}
b[4][17] =b[4][19] = b[4][21] = 220;
b[4][18] =b[4][20] = b[4][22] = 255;
b[4][23]= 220; b[4][24]= 221;
for( i=25; i<29; i++)
{
b[4][i] = 255;
}
b[4][29] = b[4][31]=b[4][33] = 220;
b[4][30] = b[4][32] = 255;
for( i=34; i<38; i++)
{
b[4][i] = 255;
}
b[4][38] = 220;
b[4][39] = b[4][41] =b[4][43]= 255;
b[4][40] =b[4][42] = b[4][44]=220;
for( i=45; i<50; i++)
{
b[4][i] = 255;
}
b[4][50] =b[4][52] =b[4][54] = 220;
b[4][56] = 220;
b[4][51] =b[4][53] =b[4][55] = 255;
for( i=57; i<61; i++)
{
b[4][i] = 255;
}
b[4][61] = 220;

for( i=62; i<67; i++)
{
b[4][i] = 255;
}
b[4][67] = 220;

for( i=68; i<80; i++)
{
b[4][i] = 255;
}
movv2(b,4);
//------------
for( i=0; i<6; i++)
{
a[3][i] = 255;
}
```

```
a[3][6] = 220;
for( i=7; i<18; i++)
{
a[3][i] = 255;
}
a[3][18] = 220;
for( i=19; i<26; i++)
{
a[3][i] = 255;
}
a[3][26] = 220;
for( i=27; i<34; i++)
{
a[3][i] = 255;
}
a[3][34] = 220;
for( i=35; i<43; i++)
{
a[3][i] = 255;
}
a[3][43] = 220;
for( i=44; i<55; i++)
{
a[3][i] = 255;
}
a[3][55] = a[3][59] = 220;
a[3][56] = a[3][60] = 220;
a[3][57] = 255;a[3][58] = 255;

for( i=62; i<80; i++)
{
a[3][i] = 255;
}
movvl(a,3);
//--------------------
for( i=0; i<6; i++)
{
b[3][i] = 255;
}
b[3][6] =220;
for( i=7; i<11; i++)
{
b[3][i] = 255;
}
b[3][11]  = 220;
for( i=12; i<17; i++)
{
b[3][i] = 255;
}
b[3][17]  = 220;
for( i=18; i<34; i++)
{
b[3][i] = 255;
}
b[3][34] = 220;
```

```
for( i=35; i<41; i++)
{
b[3][i] = 255;
}
b[3][41] = 220;
for( i=42; i<49; i++)
{
b[3][i] = 255;
}
b[3][49] = 220;

for( i=50; i<56; i++)
{
b[3][i] = 255;
}
b[3][56] = 220;

for( i=57; i<61; i++)
{
b[3][i] = 255;
}
b[3][61] =220;
for( i=62; i<65; i++)
{
b[3][i] = 255;
}
b[3][65] = b[3][67]=220;b[3][66] = 255;
for( i=68; i<80; i++)
{
b[3][i] = 255;
}
movv2(b,3);
//-----------
for( i=0; i<5; i++)
{
a[2][i] = 255;
}
a[2][5] = 220;
for( i=6; i<19; i++)
{
a[2][i] = 255;
}
for( i=19; i<25; i++)
{
a[2][i] = 220;
}
for( i=25; i<34; i++)
{
a[2][i] = 255;
}
a[2][34] = 220;
for( i=35; i<43; i++)
{
a[2][i] = 255;
}
```

```
a[2][43] = 220;a[2][44] = 255;
a[2][45] = 220;a[2][46] = 255;
a[2][47] = 220;
for( i=48; i<58; i++)
{
a[2][i] = 255;
}
a[2][58] = a[2][59] = 220;

for(i=60; i<80; i++)
{
a[2][i] = 255;
}
movvl(a,2);
//----------
for( i=0; i<6; i++)
{
b[2][i] = 255;
}
b[2][6] =220;
for( i=7; i<12; i++)
{
b[2][i] = 255;
}
b[2][12]  = 220;
for( i=13; i<17; i++)
{
b[2][i] = 255;
}
b[2][18] = b[2][20]= 255;
b[2][17] = b[2][19] =b[2][21]= 220;
for( i=22; i<29; i++)
{
b[2][i] = 255;
}
b[2][29] = 220;b[2][31] = b[2][33] =b[2][41]= 220;
b[2][30] =b[2][32] =255;
for( i=34; i<41; i++)
{
b[2][i] = 255;
}
b[2][41] = 220;
for( i=42; i<48; i++)
{
b[2][i] = 255;
}
b[2][48] = b[2][52] = 220;b[2][54] = b[2][56] = 220;
b[2][49] = b[2][50] = b[2][51]=255;
b[2][53] = b[2][55] =b[2][60]= 255;
b[2][61] = b[2][64] = b[2][67]=220;
b[2][62] =b[2][63] = 255;
b[2][65] =b[2][66] = b[2][59]=255;
for( i=68; i<80; i++)
{
b[2][i] = 255;
}
movv2(b,2);
//--------------
for( i=0; i<6; i++)
{
a[1][i] = 255;
}
a[1][6] = 220;
for( i=7; i<20; i++)
{
a[1][i] = 255;
}
a[1][20] = 220;
for( i=20; i<24; i++)
{
a[1][i] = 255;
}
a[1][24] = 220;
for( i=25; i<34; i++)
{
a[1][i] = 255;
}
a[1][34] = 220;
for( i=35; i<43; i++)
{
a[1][i] = 255;
}
a[1][43] = 220;
for( i=44; i<56; i++)
{
a[1][i] = 220;
}
a[1][56] = a[1][57] = 220;

for( i=58; i<60; i++)
{
a[1][i] = 255;
}
a[1][60] = a[1][61] = 220;

for( i=62; i<80; i++)
{
a[1][i] = 255;
}
movv1(a,1);
//------------
for( i=0; i<6; i++)
{
b[1][i] = 255;
}
b[1][6] =220;
for( i=7; i<11; i++)
{
b[1][i] = 255;
}
```

```
b[1][11]  = 220;
for( i=12; i<17; i++)
{
b[1][i] = 255;
}
b[1][17] = 220;
for( i=18; i<28; i++)
{
b[1][i] = 255;
}
b[1][28] = 220;
for( i=29; i<41; i++)
{
b[1][i] = 255;
}
b[1][41] = 220;
for( i=42; i<49; i++)
{
b[1][i] = 255;
}
b[1][49] = 220;
for( i=50; i<61; i++)
{
b[1][i] = 255;
}
b[1][61]=b[1][63]=b[1][67]= 220;
b[1][62]=b[1][64]=b[1][65]=255;
b[1][66]=255;
for( i=68; i<80; i++)
{
b[1][i] = 255;
}
movv2(b,1);
//------------
for( i=0; i<8; i++)
{
a[0][i] = 255;
}
a[0][8] = a[0][10]=220;a[0][9] = a[0][11]=255;
a[0][12] =a[0][13]=220;
for( i=13; i<21; i++)
{
a[0][i] = 255;
}
a[0][21] = 220;a[0][22] = 255;a[0][23] = 220;
for(i=24; i<30; i++)
{
a[0][i] = 255;
}
a[0][30] = 220;
for( i=31; i<37; i++)
{
a[0][i] = 255;
}
a[0][38] = 219;
```

```
for( i=39; i<43; i++)
{
a[0][i] = 255;
}
a[0][43] = 220;a[0][44] = 255;a[0][45] = 220;
a[0][46] = 255;a[0][47] = a[0][49] = 220;
a[0][50] = 220;

for( i=50; i<54; i++)
{
a[0][i] = 255;
}
a[0][54] = a[0][55] = 220;

for( i=56; i<62; i++)
{
a[0][i] = 255;
}
a[0][62] = a[0][63] = 220;

for( i=64; i<80; i++)
{
a[0][i] = 255;
}
movvl(a,0);
//---------------
for( i=0; i<6; i++)
{
b[0][i] = 255;
}
b[0][6] =220;b[0][7] =b[4][9]=255;b[0][8] =b[4][10]=220;
for( i=11; i<17; i++)
{
b[0][i] = 255;
}
b[0][17] =b[0][19] = b[0][21] = 220;
b[0][18] =b[0][20] = b[0][22] = 255;
b[0][23]= 220;b[0][24]= 221;
for( i=25; i<29; i++)
{
b[0][i] = 255;
}
b[0][29] = b[0][31]=b[0][33] = 220;
b[0][30] = b[0][32] = 255;
for( i=34; i<38; i++)
{
b[0][i] = 255;
}
b[0][38] = 220;
b[0][39] = b[0][41] =b[0][43]= 255;
b[0][40] =b[0][42] = b[0][44]=220;
for( i=45; i<50; i++)
{
b[0][i] = 255;
}
```