

# **Assignment Of Computer Graphics**

**2020 -2021**

**BTech 5<sup>th</sup> Sem**

*Dr. Dhananjoy Bhakta (CSE)*



**भारतीय सूचना प्रौद्योगिकी संस्थान राँची**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RANCHI**

(An Institution of National importance under act of Parliament)

(Ranchi - 834010), Jharkhand

**Department of Computer Science and Engineering**

# ***INDEX***

<b>Sr. No.</b>	<b>Experiment</b>	<b>Date</b>
1	National Flag using DDA Algorithm	
2	<i>Solar System using midpoint circle Algorithm</i>	
3	<i>Sky with Cloud using Eclipse Generation Algorithm</i>	
4	<i>Cloud Colouring with Fill Algorithm</i>	
5	<i>National Flag using Fill Algorithm</i>	
6	<i>Mickey Mouse shape by using DDA, and Circle generation algorithm</i>	

***By Prithwiraj Samanta(2018UGCS002R)***

# ***ASSIGNMENT NO.1***

## **Question**

Design our national flag using set of lines generated by DDA or Bresenham line drawing algorithm

## **Program**

```
#include<bits/stdc++.h>

#include<graphics.h>

using namespace std;

int main(){

    int gd=DETECT,gm;

    initgraph(&gd,&gm, NULL);

    //ORANGE RECTANGLE

    int x1=40,x2=300;

    for(int i=0;i<40;i++){

        int y1,y2;

        y1=y2=20+i;

        int dx=x2-x1;

        int dy=y2-y1;

        int p = 2*dy;
```

```

    int e = p - dx;

    int y = y1;

    for(int x=x1;x<=x2;x++){

        putpixel(x,y,12); //12 is orange

        //delay(1);

        e+=p;

        if(e>=0){

            y++;

            e-=2*dx;

        }

    }

}

//white RECTANGLE

for(int i=0;i<40;i++){

    int y1,y2;

    y1=y2=60+i;

    int dx=x2-x1;

    int dy=y2-y1;

    int p = 2*dy;

    int e = p - dx;

```

```

    int y=y1;

    for(int x=x1;x<=x2;x++){

        putpixel(x,y,WHITE);

        //    delay(1);

        e+=p;

        if(e>=0){

            y++;

            e-=2*dx;

        }

    }

}

float a=170; //center

float b=79;  //center

int r=21;    //radius

setcolor(BLUE);

circle(a,b,r);

float PI = 3.14;

//spokes

for(int i=0;i<=360;i=i+15)

{

```

```

        int x=r*cos(i*PI/180);

        int y=r*sin(i*PI/180);

        line(a,b,a+x,b-y);

    }

//GREEN RECTANGLE

for(int i=0;i<40;i++){

    int y1,y2;

    y1=y2=100+i;

    int dx=x2-x1;

    int dy=y2-y1;

    int p = 2*dy;

    int e = p - dx;

    int y =y1;

    for(int x=x1;x<=x2;x++){

        putpixel(x,y,GREEN);

        //    delay(1);

        e+=p;

        if(e>=0){

            y++;

            e-=2*dx;

```

```

    }

}

}

delay(10000);

}.

```

## Output

The screenshot shows a Linux desktop environment. On the left, a window titled "SDL-ibgraph - Graphics on GNU/Linux" displays a graphical output of the Indian flag (saffron, white, and green horizontal stripes with the Ashoka Chakra in the center). Below this window is a file explorer showing a list of files: Midpoint\_Eclipse.cpp, outtextxy.c, poly\_clipping.cpp, README.txt, and scan\_line.cpp. In the center, a code editor displays C++ code for a graphics program, including a for loop for drawing a flag. On the right, a terminal window titled "X Terminal (/main) - ~/GitHub/Coding/Competitive/Graphics - Atom" shows the execution of the program. The terminal output includes error messages from xcb and SDL, indicating a multi-threaded client issue and an assertion failure in xcb\_xlib\_threads\_sequence\_lost. The terminal prompt shows the user is prskid1000 and the current directory is ~/GitHub/Coding/Competitive/Graphics.

```

prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcbe_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcbe_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.

```

## ***ASSIGNMENT NO.2***

### **Question**

Design a solar planet system using a set of circles generated by midpoint circle generation algorithm

### **Program**

```
#include<bits/stdc++.h>

#include<graphics.h>

using namespace std;

void circlehe(int c,int r,int e,int f){

    int x=0,y;

    float d;

    y = r ; d = 1.25 -r;

    do

    {

        putpixel(e+x,f+y,c);

        putpixel(e+x,f-y,c);

        putpixel(e-x,f+y,c);

        putpixel(e-x,f-y,c);

        putpixel(e+y,f+x,c);
```



```

    putpixel(e+y,f-x,c);

    putpixel(e-y,f+x,c);

    putpixel(e-y,f-x,c);

    if(d<0)

    {

        x++;

        y= y;

        d= d+2*x+2;

        }

    else

    {

        x++;

        y--;

        d = d+2*x-2*y+1;

    }

    }while(x<y);

}

int main(){

    int gd=DETECT,gm;

```

```
detectgraph(&gd,&gm);
```

```
initgraph(&gd,&gm,NULL);
```

```
cout<<"Solar System with Circle Md Pt 32";
```

```
//sun
```

```
for(int i=0;i<40;i++){
```

```
    circlehe(14,i,400,400);
```

```
}
```

```
//9 circles
```

```
circlehe(15,90,400,400);
```

```
circlehe(15,120,400,400);
```

```
circlehe(15,150,400,400);
```

```
circlehe(15,200,400,400);
```

```
circlehe(15,250,400,400);
```

```
circlehe(15,300,400,400);
```

```
circlehe(15,340,400,400);
```

```
circlehe(15,390,400,400);
```

```
// for(int i=1;i<=8;i++){
```

```
//     circlehe(15,30*(i+2),400,400);
```

```
// }
```

```
//Mercury 22r grey color
```

```
for(int i=0;i<=21;i++)
```

```
circlehe(7,i,340,340);
```

```
//Venus 23 12 light red
```

```
for(int i=0;i<=23;i++)
```

```
circlehe(12,i,280,400);
```

```
//Earth blue=1 r=25
```

```
for(int i=0;i<=26;i++)
```

```
circlehe(1,i,295,295);
```

```
//Mars red=4 r = 22
```

```
for(int i=0;i<=22;i++)
```

```
circlehe(4,i,365,205);
```

```
//Jupiter brown 38 radius
```

```
for(int i=0;i<=38;i++)
```

```
circlehe(6,i,300,170);
```

```
//Saturn orange 35
```

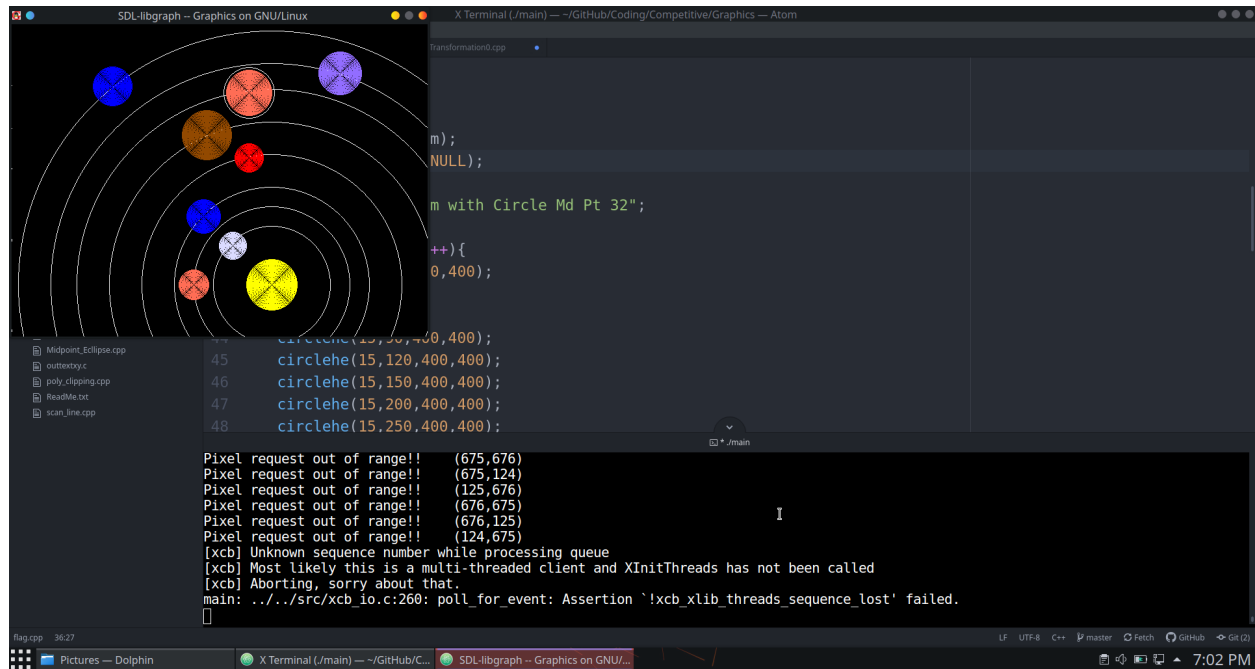
```
    for(int i=0;i<=35;i++)
        circlehe(12,i,365,105);
//    //ring
    circlehe(15,39,365,105);
//    //uranus 9 33
    for(int i=0;i<=33;i++)
        circlehe(9,i,505,75);
//    //neptune 1 30
    for(int i=0;i<=30;i++)
        circlehe(1,i,155,95);

//
    getch();

    return 0;

}
```

# Output



## ***ASSIGNMENT NO.3***

### **Question**

Design a sky consisting of clouds using set of ellipses or circles generated by midpoint ellipse generation algorithm

### **Program**

Draw a line using DDA algori

```
#include<bits/stdc++.h>
```

```
#include<graphics.h>
```

```
using namespace std;
```

```
void sky(){
```

```
    int x1=20,x2=420;
```

```
    for(int i=20;i<=420;i++){
```

```
        int y1,y2;
```

```
        y1=y2=i;
```

```
        int dx=x2-x1;
```

```
        int dy=y2-y1;
```

```
        int p = 2*dy;
```

```
        int e = p - dx;
```

```
        int y =y1;
```

```

        for(int x=x1;x<=x2;x++){

            putpixel(x,y,LIGHTBLUE);

            //    delay(1);

            e+=p;

            if(e>=0){

                y++;

                e-=2*dx;

            }

        }

    }

}

void ellipsehere(float rx,float ry,int e,int f){

    float x=0,y=ry;

    float d1=(ry*ry) - (rx*rx*ry) + (0.25*rx*rx);

    float dy =2*rx*rx*y, dx=2*ry*ry*x;

    do{

        putpixel(e+x,y+f,WHITE);

        putpixel(e-x,y+f,WHITE);

        putpixel(e+x,f-y,WHITE);

        putpixel(e-x,f-y,WHITE);

```

```

    if(d1<0){

        x=x+1;

        dx+=2*ry*ry;

        d1=d1+dx+ry*ry;

    }

    else{

        x=x+1;

        y=y-1;

        dx+=2*ry*ry;

        dy-=2*rx*rx;

        d1=d1+dx-dy+ry*ry;

    }

} while(dx<dy);

float d2= ((ry * ry) * ((x + 0.5) * (x + 0.5))) + ((rx * rx) * ((y - 1) * (y - 1)))
- (rx * rx * ry * ry);

do{

    putpixel(e+x,y+f,WHITE);

    putpixel(e-x,y+f,WHITE);

    putpixel(e+x,-y+f,WHITE);

    putpixel(e-x,-y+f,WHITE);

```



```

        if(d2>0){

            y=y-1;

            dy-=2*rx*rx;

            d2=d2-dy+rx*rx;

        }

        else{

            x=x+1;

            y=y-1;

            dx+=2*ry*ry;

            dy-=2*rx*rx;

            d2=d2+dx-dy+rx*rx;

        }

    } while(y>0);

}

void circlehe(int c,int r,int e,int f){

    int x=0,y;

    float d;

    y = r ; d = 1.25 -r;

    do

    {

```

```

putpixel(e+x,f+y,c);

putpixel(e+x,f-y,c);

putpixel(e-x,f+y,c);

putpixel(e-x,f-y,c);

putpixel(e+y,f+x,c);

putpixel(e+y,f-x,c);

putpixel(e-y,f+x,c);

putpixel(e-y,f-x,c);

if(d<0)

{

    x++;

    y= y;

    d= d+2*x+2;

}

else

{

    x++;

    y--;

    d = d+2*x-2*y+1;

}

```

```

        }while(x<y);
    }

int main(){

    int gd=DETECT,gm;

    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,NULL);


    sky();

    cout<<"Clouds with Circle and Ellipse Md Pt 32";

    //cloud 1
    for(int i=0;i<20;i++){

        circlehe(15,i,100,85);

    }

    for(int i=0;i<15;i++){

        circlehe(15,i,135,90);

    }

    for(int i=0;i<15;i++){

        circlehe(15,i,65,90);

    }

```

```
int i=60,j=20;

for(;i>=0&& j>=0;i--,j--)

ellipsehere(i,j,100,100);
```

```
//cloud 2
```

```
for(int i=0;i<20;i++){

    circlehe(15,i,200,175);

}
```

```
for(int i=0;i<20;i++){

    circlehe(15,i,160,185);

}
```

```
for(int i=0;i<20;i++){

    circlehe(15,i,240,185);

}
```

```
i=80,j=30;

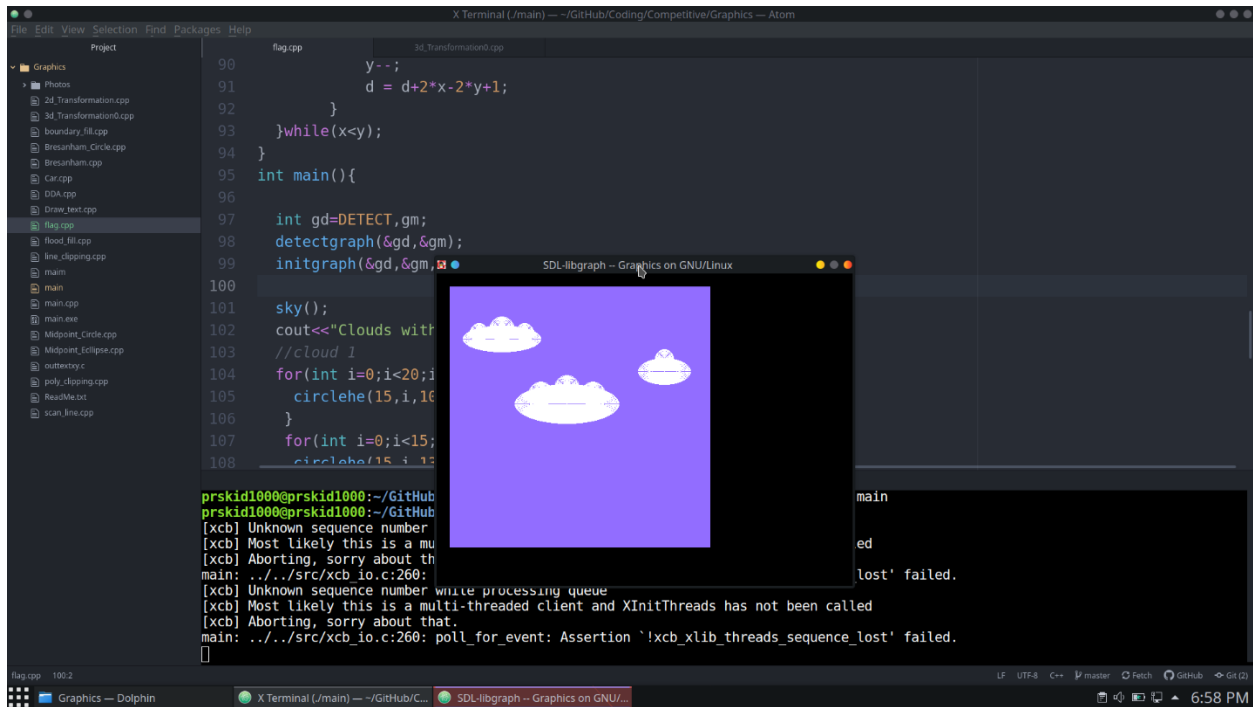
for(;i>=0&& j>=0;i--,j--)

    ellipsehere(i,j,200,200);
```

```
//cloud 3
```

```
for(int i=0;i<15;i++){  
    circlehe(15,i,350,130);  
}  
  
i=40,j=20;  
  
for(;i>=0&& j>=0;i--,j--)  
    ellipsehere(i,j,350,150);  
  
getch();  
  
return 0;  
}
```

# Output



The screenshot shows an IDE with a project named 'Graphics' on the left. The main editor displays the code for 'flag.cpp'. The code includes a loop for calculating a value 'd' and a main function that initializes a graphics window and draws clouds. A separate window titled 'SDL-tlbgraph -- Graphics on GNU/Linux' shows the graphical output: a black background with three white, stylized clouds. The terminal at the bottom shows the execution of the program, which ends with an assertion failure: 'Assertion '!xcb\_xlib\_threads\_sequence\_lost' failed.'

```
flag.cpp
90     y--;
91     d = d+2*x-2*y+1;
92 }
93 }while(x<y);
94 }
95 int main(){
96
97     int gd=DETECT,gm;
98     detectgraph(&gd,&gm);
99     initgraph(&gd,&gm,
100
101     sky();
102     cout<<"Clouds with
103     //cloud 1
104     for(int i=0;i<20;i
105         circlehe(15,i,10
106     }
107     for(int i=0;i<15;
108         circlehe(15,i,10

prskid1000@prskid1000:~/GitHub
prskid1000@prskid1000:~/GitHub
[xcb] Unknown sequence number
[xcb] Most likely this is a mu
[xcb] Aborting, sorry about th
main: ../../src/xcb_io.c:260:
[xcb] Unknown sequence number write processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
[]

main
ed
lost' failed.
```

# ***ASSIGNMENT NO.4***

## **Question**

Draw 3 clouds by using graphics functions and fill the 3 clouds with saphron, white and green colors by using flood fill Algorithm.

## **Program**

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void floodfillalgo(int x,int y,int c)
{
    if(getpixel(x,y)==c)
        return;
    putpixel(x,y,c);
    floodfillalgo(x+1,y,c);
    floodfillalgo(x-1,y,c);
    floodfillalgo(x,y+1,c);
    floodfillalgo(x,y-1,c);
}
int main()
{
    int gd=DETECT,gm;
    detectgraph(&gd,&gm);
```

```
initgraph(&gd,&gm,NULL);

//Saffron Cloud
setcolor(12);
circle(100,85,20);
ellipse(100, 100, 0, 360, 50, 25);
floodfillalgo(90,80,12);
floodfillalgo(90,70,12);
floodfillalgo(80,106,12);

//White Cloud
setcolor(15);
circle(205,85,20);
ellipse(205,100,0,360,50,25);
floodfillalgo(205,80,15);
floodfillalgo(205,70,15);
floodfillalgo(205,106,15);

//Green Cloud
setcolor(GREEN);
circle(310,85,20);
ellipse(310,100,0,360,50,25);
floodfillalgo(310,80,GREEN);
floodfillalgo(310,70,GREEN);
floodfillalgo(310,106,GREEN);
getch();
closegraph();
```



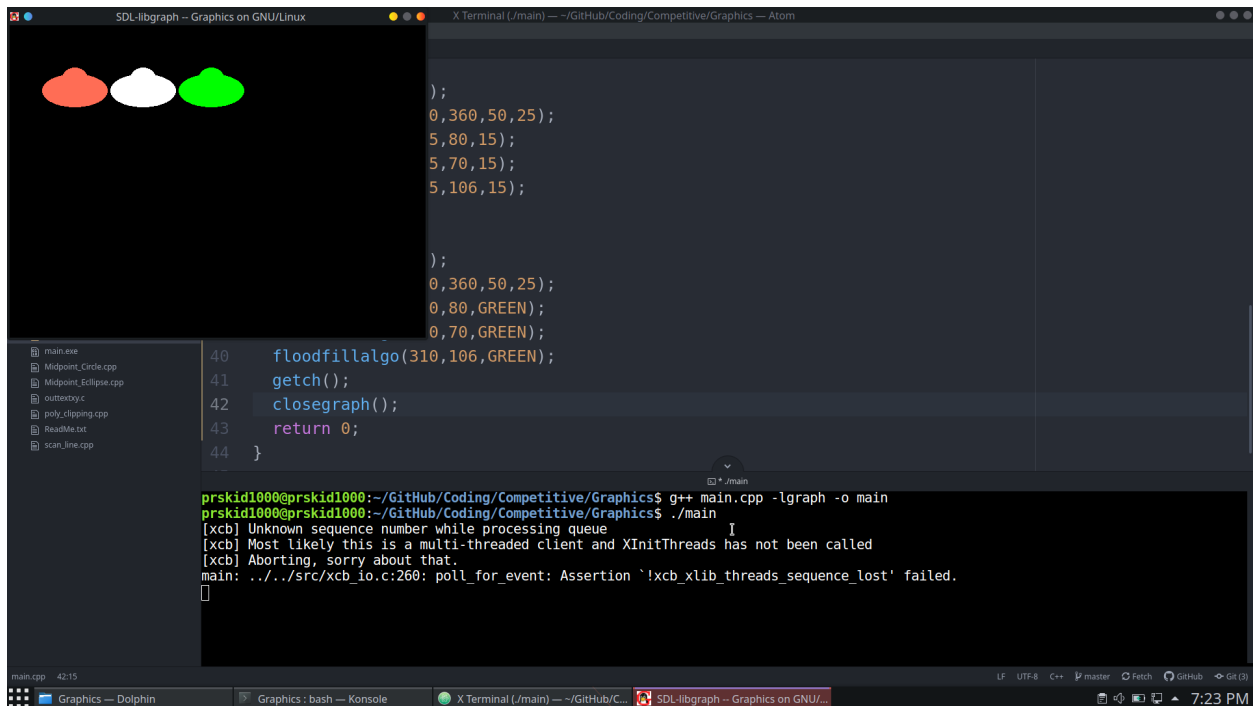
```

return 0;

}

```

## Output



```

);
0,360,50,25);
5,80,15);
5,70,15);
5,106,15);

);
0,360,50,25);
0,80,GREEN);
0,70,GREEN);

40 floodfillalgo(310,106,GREEN);
41 getch();
42 closegraph();
43 return 0;
44 }

prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ g++ main.cpp -lgraph -o main
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.

```

# ***ASSIGNMENT NO.5***

## **Question**

Draw our national flag by using graphics functions and fill appropriate colors by using boundary fill algorithm

## **Program**

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void boundaryfill(int x,int y,int boundarycolor,int newcolor)
{
    if(getpixel(x,y)==boundarycolor || getpixel(x,y)==newcolor) return;
    putpixel(x,y,newcolor);
    boundaryfill(x+1,y,boundarycolor,newcolor);
    boundaryfill(x-1,y,boundarycolor,newcolor);
    boundaryfill(x,y+1,boundarycolor,newcolor);
    boundaryfill(x,y-1,boundarycolor,newcolor);
}
int main()
{
    int gd=DETECT,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,NULL);
```

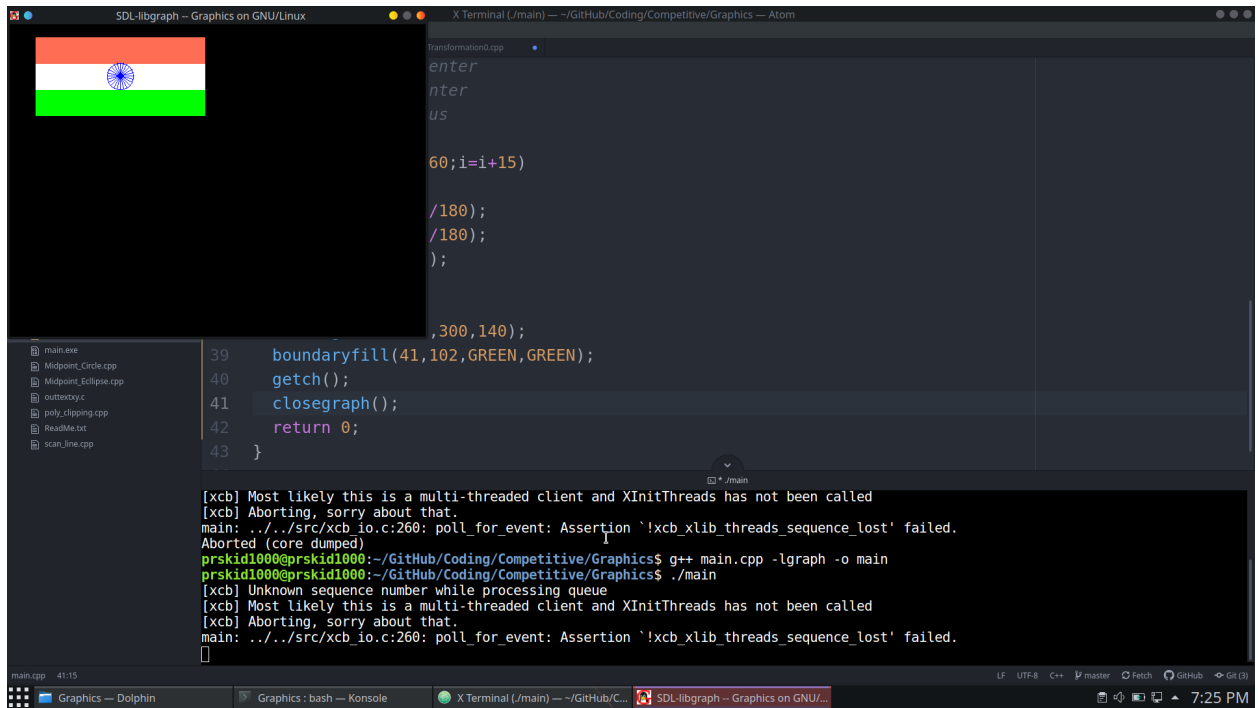
```

setcolor(LIGHTRED);
rectangle(40,20,300,60);
boundaryfill(41,21,LIGHTRED,LIGHTRED);
setcolor(WHITE);
rectangle(40,61,300,100);
boundaryfill(41,62,WHITE,WHITE);
setcolor(BLUE);
circle(170,80,20);
float a=170; //center
float b=79; //center
int r=21; //radius
float PI = 3.14;
for(int i=0;i<=360;i=i+15)
{
int x=r*cos(i*PI/180);
int y=r*sin(i*PI/180);
line(a,b,a+x,b-y);
}
setcolor(GREEN);
rectangle(40,101,300,140);
boundaryfill(41,102,GREEN,GREEN);
getch();
closegraph();
return 0;

```

}

## Output



The screenshot shows a code editor with a dark theme. The top window displays a C++ program with the following code:

```
Transformation0.cpp
enter
nter
us

60;i=i+15)

/180);
/180);
);

,300,140);

39 boundaryfill(41,102,GREEN,GREEN);
40 getch();
41 closegraph();
42 return 0;
43 }
```

The bottom window shows the output of the program, which includes several error messages from the XCB library:

```
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ g++ main.cpp -lgraph -o main
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
```

The left sidebar of the code editor shows a file explorer with the following files:

- main.exe
- Midpoint\_Circle.cpp
- Midpoint\_Ellipse.cpp
- outtextxy.c
- poly\_clipping.cpp
- ReadMe.txt
- scan\_line.cpp

The bottom status bar of the code editor shows the following information:

- main.cpp 41:15
- Graphics — Dolphin
- Graphics: bash — Konsole
- X Terminal (/main) — ~/GitHub/C...
- SDL-Ilbgraph — Graphics on GNU/Linux
- LF UTF-8 C++ master Fetch GitHub Git (B)
- 7:25 PM

# ***ASSIGNMENT NO.6***

## **Question**

Draw a Mickey Mouse shape by using DDA, circle generation algorithm

## **Program**

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void boundaryfill(int x,int y,int boundarycolor,int newcolor){
    if(getpixel(x,y)==boundarycolor || getpixel(x,y)==newcolor) return;
    putpixel(x,y,newcolor);
    boundaryfill(x+1,y,boundarycolor,newcolor);
    boundaryfill(x-1,y,boundarycolor,newcolor);
    boundaryfill(x,y+1,boundarycolor,newcolor);
    boundaryfill(x,y-1,boundarycolor,newcolor);
}
void background(){
    int x1=20,x2=600;
    for(int i=20;i<600;i++){
        int y1,y2;
        y1=y2=i;
        int dx=x2-x1;
        int dy=y2-y1;
```

```

    int steps;
    if(abs(dx)>abs(dy)) steps = abs(dx);
    else steps = abs(dy);
    int xinc = dx/steps;
    int yinc = dy/steps;
    int x=0,y=i;
    x+=xinc;
    y+=yinc;
    for(int i=0;i<steps;i++){
        putpixel(x,y,WHITE);
        //delay(1);
        x+=xinc;
        y+=yinc;
    }
}

}

void ellipsehere(float rx,float ry,int e,int f,int color){
    float x=0,y=ry;
    float d1=(ry*ry) - (rx*rx*ry) + (0.25*rx*rx);
    float dy =2*rx*rx*y, dx=2*ry*ry*x;
    do{
        putpixel(e+x,y+f,color);
        putpixel(e-x,y+f,color);
        putpixel(e+x,f-y,color);
        putpixel(e-x,f-y,color);
    }
}

```

```

    if(d1<0){
        x=x+1;
        dx+=2*ry*ry;
        d1=d1+dx+ry*ry;
    }
    else{
        x=x+1;
        y=y-1;
        dx+=2*ry*ry;
        dy-=2*rx*rx;
        d1=d1+dx-dy+ry*ry;
    }
} while(dx<dy);

float d2= ((ry * ry) * ((x + 0.5) * (x + 0.5))) + ((rx * rx) * ((y - 1) * (y - 1)))
- (rx * rx * ry * ry);

do{
    putpixel(e+x,y+f,color);
    putpixel(e-x,y+f,color);
    putpixel(e+x,-y+f,color);
    putpixel(e-x,-y+f,color);
    if(d2>0){
        y=y-1;
        dy-=2*rx*rx;
        d2=d2-dy+rx*rx;
    }
}

```

```

        else{
            x=x+1;
            y=y-1;
            dx+=2*ry*ry;
            dy-=2*rx*rx;
            d2=d2+dx-dy+rx*rx;
        }
    } while(y>0);
}

void circlehe(int r,int e,int f,int c){
    int x=0,y;
    float d;
    y = r ; d = 1.5 -r;
    do
    {
        putpixel(e+x,f+y,c);
        putpixel(e+x,f-y,c);
        putpixel(e-x,f+y,c);
        putpixel(e-x,f-y,c);
        putpixel(e+y,f+x,c);
        putpixel(e+y,f-x,c);
        putpixel(e-y,f+x,c);
        putpixel(e-y,f-x,c);
        if(d<0)
        {

```



```

        x++;
        y= y;
        d= d+2*x+2;
    }
else
{
    x++;
    y--;
    d = d+2*x-2*y+1;
}
}while(x<y);
}

int main(){
    int gd=DETECT,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,NULL);
        background();
        circlehe(60,100,100,BLACK);
        boundaryfill(80,80,BLACK,BLACK);
        circlehe(60,300,100,BLACK);
        boundaryfill(280,80,BLACK,BLACK);
        circlehe(120,200,200,BLACK);
        boundaryfill(180,180,BLACK,BLACK);
        delay(100);
        getch();

```

```

    return 0;
}

```

## Output

The screenshot shows an IDE with a project named 'Graphics'. The file explorer on the left lists various source files. The main editor displays the code for 'main.cpp', which defines a circle and draws it using the Bresenham's circle algorithm. The code includes comments for each step of the algorithm. The output window at the bottom shows a series of 'Pixel request out of range!!' messages, indicating that the algorithm is attempting to draw pixels outside the defined circle's bounding box. A small window titled 'SDL-ibgraph -- Graphics on GNU/Linux' displays a black circle on a white background, which is the visual output of the program.

```

main.cpp
78  int x=0,y;
79  float d;
80  y = r ; d = 1.5 -r;
81  do
82  {
83      putpixel(e+x,f+y,c);
84      putpixel(e+x,f-y,c);
85      putpixel(e-x,f+y,c);
86      putpixel(e-x,f-y,c);
87      putpixel(e+y,f+x,c);
88      putpixel(e+y,f-x,c);
89      putpixel(e-y,f+x,c);
90      putpixel(e-y,f-x,c);
91      if(d<0)
92      {
93          x++;
94          y= y;

```

```

Pixel request out of range!! (571,599)
Pixel request out of range!! (572,599)
Pixel request out of range!! (573,599)
Pixel request out of range!! (574,599)
Pixel request out of range!! (575,599)
Pixel request out of range!! (576,599)
Pixel request out of range!! (577,599)
Pixel request out of range!! (578,599)
Pixel request out of range!! (579,599)
Pixel request out of range!! (580,599)

```