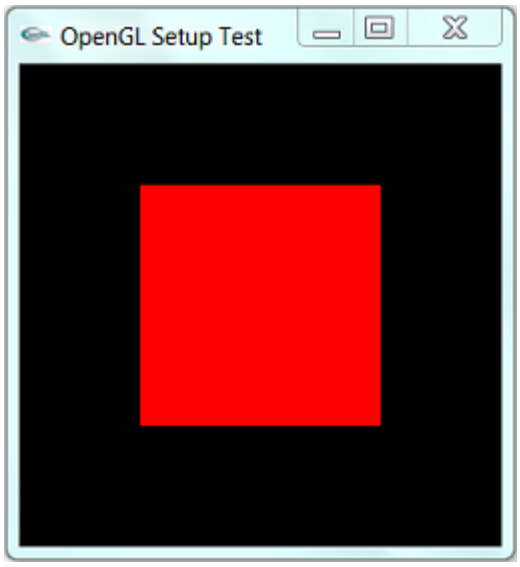
Practical -14

2D Graphics functions:

```
/*
 * GL01Hello.cpp: Test OpenGL C/C++ Setup
 */
#include <windows.h>  // For MS Windows
#include <GL/glut.h>  // GLUT, includes glu.h and gl.h

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
void display() {
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
   glClear(GL_COLOR_BUFFER_BIT);         // Clear the color buffer

   // Draw a Red 1x1 Square centered at origin
   glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
      glColor3f(1.0f, 0.0f, 0.0f); // Red
      glVertex2f(-0.5f, -0.5f);    // x, y
      glVertex2f( 0.5f, -0.5f);
      glVertex2f( 0.5f,  0.5f);
      glVertex2f(-0.5f,  0.5f);
   glEnd();

   glFlush();  // Render now
}

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
   glutInit(&argc, argv);                 // Initialize GLUT
   glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title
   glutInitWindowSize(320, 320);   // Set the window's initial width & height
   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
   glutDisplayFunc(display); // Register display callback handler for window re-paint
```

```
  glutMainLoop();          // Enter the infinitely event-processing loop
  return 0;
}
```



3D Graphics functions:

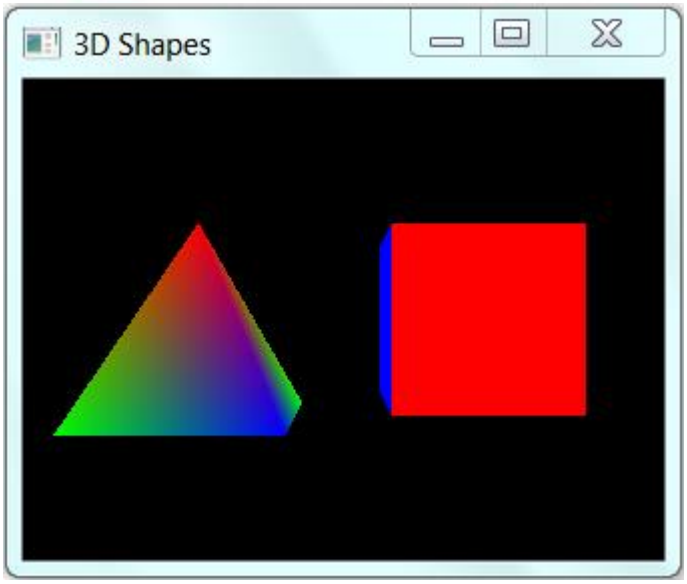
```
/*
 * OGL01Shape3D.cpp: 3D Shapes
 */
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Global variables */
char title[] = "3D Shapes";

/* Initialize OpenGL Graphics */
void initGL() {
  glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
  glClearDepth(1.0f);                   // Set background depth to farthest
  glEnable(GL_DEPTH_TEST);   // Enable depth testing for z-culling
  glDepthFunc(GL_LEQUAL);    // Set the type of depth-test
  glShadeModel(GL_SMOOTH);   // Enable smooth shading
```

```
  glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);  // Nice perspective
corrections
}

/* Handler for window-repaint event. Called back when the window first appears and
   whenever the window needs to be re-painted. */
void display() {
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear color and
depth buffers
  glMatrixMode(GL_MODELVIEW);     // To operate on model-view matrix

  // Render a color-cube consisting of 6 quads with different colors
  glLoadIdentity();                 // Reset the model-view matrix
  glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen

  glBegin(GL_QUADS);                // Begin drawing the color cube with 6 quads
    // Top face (y = 1.0f)
    // Define vertices in counter-clockwise (CCW) order with normal pointing out
    glColor3f(0.0f, 1.0f, 0.0f);     // Green
    glVertex3f( 1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f,  1.0f);
    glVertex3f( 1.0f, 1.0f,  1.0f);

    // Bottom face (y = -1.0f)
    glColor3f(1.0f, 0.5f, 0.0f);     // Orange
    glVertex3f( 1.0f, -1.0f,  1.0f);
    glVertex3f(-1.0f, -1.0f,  1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);

    // Front face  (z = 1.0f)
    glColor3f(1.0f, 0.0f, 0.0f);     // Red
    glVertex3f( 1.0f,  1.0f, 1.0f);
    glVertex3f(-1.0f,  1.0f, 1.0f);
```

```
   glVertex3f(-1.0f, -1.0f, 1.0f);
   glVertex3f( 1.0f, -1.0f, 1.0f);

   // Back face (z = -1.0f)
   glColor3f(1.0f, 1.0f, 0.0f);     // Yellow
   glVertex3f( 1.0f, -1.0f, -1.0f);
   glVertex3f(-1.0f, -1.0f, -1.0f);
   glVertex3f(-1.0f,  1.0f, -1.0f);
   glVertex3f( 1.0f,  1.0f, -1.0f);

   // Left face (x = -1.0f)
   glColor3f(0.0f, 0.0f, 1.0f);     // Blue
   glVertex3f(-1.0f,  1.0f,  1.0f);
   glVertex3f(-1.0f,  1.0f, -1.0f);
   glVertex3f(-1.0f, -1.0f, -1.0f);
   glVertex3f(-1.0f, -1.0f,  1.0f);

   // Right face (x = 1.0f)
   glColor3f(1.0f, 0.0f, 1.0f);     // Magenta
   glVertex3f(1.0f,  1.0f, -1.0f);
   glVertex3f(1.0f,  1.0f,  1.0f);
   glVertex3f(1.0f, -1.0f,  1.0f);
   glVertex3f(1.0f, -1.0f, -1.0f);
glEnd();  // End of drawing color-cube

// Render a pyramid consists of 4 triangles
glLoadIdentity();                 // Reset the model-view matrix
glTranslatef(-1.5f, 0.0f, -6.0f);  // Move left and into the screen

glBegin(GL_TRIANGLES);           // Begin drawing the pyramid with 4 triangles
   // Front
   glColor3f(1.0f, 0.0f, 0.0f);     // Red
   glVertex3f( 0.0f, 1.0f, 0.0f);
   glColor3f(0.0f, 1.0f, 0.0f);     // Green
   glVertex3f(-1.0f, -1.0f, 1.0f);
```

```
    glColor3f(0.0f, 0.0f, 1.0f);     // Blue
    glVertex3f(1.0f, -1.0f, 1.0f);

    // Right
    glColor3f(1.0f, 0.0f, 0.0f);     // Red
    glVertex3f(0.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);     // Blue
    glVertex3f(1.0f, -1.0f, 1.0f);
    glColor3f(0.0f, 1.0f, 0.0f);     // Green
    glVertex3f(1.0f, -1.0f, -1.0f);

    // Back
    glColor3f(1.0f, 0.0f, 0.0f);     // Red
    glVertex3f(0.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);     // Green
    glVertex3f(1.0f, -1.0f, -1.0f);
    glColor3f(0.0f, 0.0f, 1.0f);     // Blue
    glVertex3f(-1.0f, -1.0f, -1.0f);

    // Left
    glColor3f(1.0f,0.0f,0.0f);       // Red
    glVertex3f( 0.0f, 1.0f, 0.0f);
    glColor3f(0.0f,0.0f,1.0f);       // Blue
    glVertex3f(-1.0f,-1.0f,-1.0f);
    glColor3f(0.0f,1.0f,0.0f);       // Green
    glVertex3f(-1.0f,-1.0f, 1.0f);
  glEnd();   // Done drawing the pyramid

  glutSwapBuffers();  // Swap the front and back frame buffers (double buffering)
}

/* Handler for window re-size event. Called back when the window first appears and
   whenever the window is re-sized with its new width and height */
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
  // Compute aspect ratio of the new window
```

```
  if (height == 0) height = 1;                // To prevent divide by 0
  GLfloat aspect = (GLfloat)width / (GLfloat)height;

  // Set the viewport to cover the new window
  glViewport(0, 0, width, height);

  // Set the aspect ratio of the clipping volume to match the viewport
  glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
  glLoadIdentity();             // Reset
  // Enable perspective projection with fovy, aspect, zNear and zFar
  gluPerspective(45.0f, aspect, 0.1f, 100.0f);
}

/* Main function: GLUT runs as a console application starting at main() */
int main(int argc, char** argv) {
  glutInit(&argc, argv);           // Initialize GLUT
  glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
  glutInitWindowSize(640, 480);   // Set the window's initial width & height
  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
  glutCreateWindow(title);          // Create window with the given title
  glutDisplayFunc(display);       // Register callback handler for window re-paint event
  glutReshapeFunc(reshape);       // Register callback handler for window re-size event
  initGL();                       // Our own OpenGL initialization
  glutMainLoop();                 // Enter the infinite event-processing loop
  return 0;
}
```



GLUT Setup - main()

The program contains a initGL(), display() and reshape() functions.

The main() program:

glutInit(&argc, argv);

Initializes the GLUT.

glutInitWindowSize(640, 480);

glutInitWindowPosition(50, 50);

glutCreateWindow(title);

Creates a window with a title, initial width and height positioned at initial top-left corner.

glutDisplayFunc(display);

Registers display() as the re-paint event handler. That is, the graphics sub-system calls back display() when the window first appears and whenever there is a re-paint request.

glutReshapeFunc(reshape);

Registers reshape() as the re-sized event handler. That is, the graphics sub-system calls back reshape() when the window first appears and whenever the window is re-sized.

glutInitDisplayMode(GLUT_DOUBLE);

Enables double buffering. In display(), we use glutSwapBuffers() to signal to the GPU to swap the front-buffer and back-buffer during the next VSync (Vertical Synchronization).

initGL();

Invokes the initGL() once to perform all one-time initialization tasks.

glutMainLoop();

Finally, enters the event-processing loop.

One-Time Initialization Operations - initGL()

The initGL() function performs the one-time initialization tasks. It is invoked from main() once (and only once).

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
glClearDepth(1.0f); // Set background depth to farthest
// In display()
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Set the clearing (background) color to black (R=0, G=0, B=0) and opaque (A=1), and the clearing (background) depth to the farthest (Z=1). In display(), we invoke glClear() to clear the color and depth buffer, with the clearing color and depth, before rendering the graphics. (Besides the color buffer and depth buffer, OpenGL also maintains an accumulation buffer and a stencil buffer which shall be discussed later.)

```
glEnable(GL_DEPTH_TEST); // Enable depth testing for z-culling
glDepthFunc(GL_LEQUAL); // Set the type of depth-test
```

We need to enable depth-test to remove the hidden surface, and set the function used for the depth test.

```
glShadeModel(GL_SMOOTH); // Enable smooth shading
```

We enable smooth shading in color transition. The alternative is GL_FLAT. Try it out and see the difference.

```
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST); // Nice perspective corrections
```

In graphics rendering, there is often a trade-off between processing speed and visual quality. We can use glHint() to decide on the trade-off. In this case, we ask for the best perspective correction, which may involve more processing. The default is GL_DONT_CARE.

Defining the Color-cube and Pyramid

OpenGL's object is made up of primitives (such as triangle, quad, polygon, point and line). A

primitive is defined via one or more vertices. The color-cube is made up of 6 quads. Each quad is made up of 4 vertices, defined in counter-clockwise (CCW) order, such as the normal vector is pointing out, indicating the front face. All the 4 vertices have the same color. The color-cube is defined in its local space (called model space) with origin at the center of the cube with sides of 2 units.

Similarly, the pyramid is made up of 4 triangles (without the base). Each triangle is made up of 3 vertices, defined in CCW order. The 5 vertices of the pyramid are assigned different colors. The color of the triangles are interpolated (and blend smoothly) from its 3 vertices. Again, the pyramid is defined in its local space with origin at the center of the pyramid.

PRACTICAL SET-15

OEP

```
#include<iostream>
#include<windows.h>
#include<conio.h>
#include<fstream>
#include<string.h>
#include<cstdio>
#include<cstdlib>
#include<iomanip>
using namespace std;
//global variable declaration
int k=7,r=0,flag=0;
COORD coord = {0, 0};

void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
struct date
{
    int mm,dd,yy;
};

ofstream fout;
ifstream fin;

class item
{
    int itemno;
    char name[25];
    date d;
public:
    void add()
    {
        cout<<"\n\n\tItem No: ";
        cin>>itemno;
        cout<<"\n\n\tName of the item: ";
        cin>>name;
//gets(name);
        cout<<"\n\n\tManufacturing Date(dd-mm-yy): ";
```

```
        cin>>d.mm>>d.dd>>d.yy;
    }
    void show()
    {
        cout<<"\n\tItem No: ";
        cout<<itemno;
        cout<<"\n\n\tName of the item: ";
        cout<<name;
        cout<<"\n\n\tDate : ";
        cout<<d.mm<<"-"<<d.dd<<"-"<<d.yy;
    }
    void report()
    {
        gotoxy(3,k);
        cout<<itemno;
        gotoxy(13,k);
        puts(name);
    }
    int retno()
    {
        return(itemno);

    }

};

class amount: public item
{
    float price,qty,tax,gross,dis,netamt;
public:
    void add();
    void show();
    void report();
    void calculate();
    void pay();
    float retnetamt()
    {
        return(netamt);
    }
} amt;

void amount::add()
{
    item::add();
    cout<<"\n\n\tPrice: ";
    cin>>price;
```

```
    cout<<"\n\n\tQuantity: ";
    cin>>qty;
    cout<<"\n\n\tTax percent: ";
    cin>>tax;
    cout<<"\n\n\tDiscount percent: ";
    cin>>dis;
    calculate();
    fout.write((char *)&amt,sizeof(amt));
    fout.close();
}
void amount::calculate()
{
    gross=price+(price*(tax/100));
    netamt=qty*(gross-(gross*(dis/100)));
}
void amount::show()
{
    fin.open("itemstore.dat",ios::binary);
    fin.read((char*)&amt,sizeof(amt));
    item::show();
    cout<<"\n\n\tNet amount: ";
    cout<<netamt;
    fin.close();
}

void amount::report()
{
    item::report();
    gotoxy(23,k);
    cout<<price;
    gotoxy(33,k);
    cout<<qty;
    gotoxy(44,k);
    cout<<tax;
    gotoxy(52,k);
    cout<<dis;
    gotoxy(64,k);
    cout<<netamt;
    k=k+1;
    if(k==50)
    {
        gotoxy(25,50);
        cout<<"PRESS ANY KEY TO CONTINUE...";
        getch();
        k=7;
        system("cls");
```

```
        gotoxy(30,3);
        cout<<" ITEM DETAILS ";
        gotoxy(3,5);
        cout<<"NUMBER";
        gotoxy(13,5);
        cout<<"NAME";
        gotoxy(23,5);
        cout<<"PRICE";
        gotoxy(33,5);
        cout<<"QUANTITY";
        gotoxy(44,5);
        cout<<"TAX";
        gotoxy(52,5);
        cout<<"DEDUCTION";
        gotoxy(64,5);
        cout<<"NET AMOUNT";
    }
}

void amount::pay()
{
    show();
    cout<<"\n\n\n\t\t*******************************************";
    cout<<"\n\t\t                DETAILS                  ";
    cout<<"\n\t\t*******************************************";
    cout<<"\n\n\t\tPRICE                    :"<<price;
    cout<<"\n\n\t\tQUANTITY                 :"<<qty;
    cout<<"\n\t\tTAX PERCENTAGE           :"<<tax;
    cout<<"\n\t\tDISCOUNT PERCENTAGE      :"<<dis;
    cout<<"\n\n\n\t\tNET AMOUNT               :Rs."<<netamt;
    cout<<"\n\t\t*******************************************";
}

int main()
{
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout<<setprecision(2);
    fstream tmp("temp.dat",ios::binary|ios::out);
menu:
    system("cls");
    gotoxy(25,2);
    cout<<"Super Market Billing ";
    gotoxy(25,3);
    cout<<"==========================\n\n";
    cout<<"\n\t\t1.Bill Report\n\n";
```

```
    cout<<"\t\t2.Add/Remove/Edit Item\n\n";
    cout<<"\t\t3.Show Item Details\n\n";
    cout<<"\t\t4.Exit\n\n";
    cout<<"\t\tPlease Enter Required Option: ";
    int ch,ff;
    float gtotal;
    cin>>ch;
    switch(ch)
    {
    case 1:
ss:
        system("cls");
        gotoxy(25,2);
        cout<<"Bill Details";
        gotoxy(25,3);
        cout<<"================\n\n";
        cout<<"\n\t\t1.All Items\n\n";
        cout<<"\t\t2.Back to Main menu\n\n";
        cout<<"\t\tPlease Enter Required Option: ";
        int cho;
        cin>>cho;
        if(cho==1)
        {
            system("cls");
            gotoxy(30,3);
            cout<<" BILL DETAILS ";
            gotoxy(3,5);
            cout<<"ITEM NO";
            gotoxy(13,5);
            cout<<"NAME";
            gotoxy(23,5);
            cout<<"PRICE";
            gotoxy(33,5);
            cout<<"QUANTITY";
            gotoxy(44,5);
            cout<<"TAX %";
            gotoxy(52,5);
            cout<<"DISCOUNT %";
            gotoxy(64,5);
            cout<<"NET AMOUNT";
            fin.open("itemstore.dat",ios::binary);
            if(!fin)
            {
                cout<<"\n\nFile Not Found...";
                goto menu;
            }
```

```
            fin.seekg(0);
            gtotal=0;
            while(!fin.eof())
            {
                fin.read((char*)&amt,sizeof(amt));
                if(!fin.eof())
                {
                    amt.report();
                    gtotal+=amt.retnetamt();
                    ff=0;
                }
                if(ff!=0) gtotal=0;
            }
            gotoxy(17,k);
            cout<<"\n\n\n\t\t\tGrand Total="<<gtotal;
            getch();
            fin.close();
        }
        if(cho==2)
        {
            goto menu;
        }
        goto ss;
    case 2:
db:
        system("cls");
        gotoxy(25,2);
        cout<<"Bill Editor";
        gotoxy(25,3);
        cout<<"=================\n\n";
        cout<<"\n\t\t1.Add Item Details\n\n";
        cout<<"\t\t2.Edit Item Details\n\n";
        cout<<"\t\t3.Delete Item Details\n\n";
        cout<<"\t\t4.Back to Main Menu ";
        int apc;
        cin>>apc;
        switch(apc)
        {
        case 1:
            fout.open("itemstore.dat",ios::binary|ios::app);
            amt.add();
            cout<<"\n\t\tItem Added Successfully!";
            getch();
            goto db;

        case 2:
```

```
        int ino;
        flag=0;
        cout<<"\n\n\tEnter Item Number to be Edited :";
        cin>>ino;
        fin.open("itemstore.dat",ios::binary);
        fout.open("itemstore.dat",ios::binary|ios::app);
        if(!fin)
        {
            cout<<"\n\nFile Not Found...";
            goto menu;
        }
        fin.seekg(0);
        r=0;
        while(!fin.eof())
        {
            fin.read((char*)&amt,sizeof(amt));
            if(!fin.eof())
            {
                int x=amt.item::retno();
                if(x==ino)
                {
                    flag=1;
                    fout.seekp(r*sizeof(amt));
                    system("cls");
                    cout<<"\n\t\tCurrent Details are\n";
                    amt.show();
                    cout<<"\n\n\t\tEnter New Details\n";
                    amt.add();
                    cout<<"\n\t\tItem Details editted";
                }
            }
            r++;
        }
        if(flag==0)
        {
            cout<<"\n\t\tItem No does not exist...Please Retry!";
            getch();
            goto db;
        }
        fin.close();
        getch();
        goto db;

    case 3:
        flag=0;
        cout<<"\n\n\tEnter Item Number to be deleted :";
```

```
            cin>>ino;
            fin.open("itemstore.dat",ios::binary);
            if(!fin)
            {
                cout<<"\n\nFile Not Found...";
                goto menu;
            }
//fstream tmp("temp.dat",ios::binary|ios::out);
            fin.seekg(0);
            while(fin.read((char*)&amt, sizeof(amt)))
            {
                int x=amt.item::retno();
                if(x!=ino)
                    tmp.write((char*)&amt,sizeof(amt));
                else
                {
                    flag=1;
                }
            }
            fin.close();
            tmp.close();
            fout.open("itemstore.dat",ios::trunc|ios::binary);
            fout.seekp(0);
            tmp.open("temp.dat",ios::binary|ios::in);
            if(!tmp)
            {
                cout<<"Error in File";
                goto db;
            }
            while(tmp.read((char*)&amt,sizeof(amt)))
                fout.write((char*)&amt,sizeof(amt));
            tmp.close();
            fout.close();
            if(flag==1)
                cout<<"\n\t\tItem Succesfully Deleted";
            else if (flag==0)
                cout<<"\n\t\tItem does not Exist! Please Retry";
            getch();
            goto db;
        case 4:
            goto menu;
        default:
            cout<<"\n\n\t\tWrong Choice!!! Retry";
            getch();
            goto db;
        }
```

```
    case 3:
        system("cls");
        flag=0;
        int ino;
        cout<<"\n\n\t\tEnter Item Number :";
        cin>>ino;
        fin.open("itemstore.dat",ios::binary);
        if(!fin)
        {
            cout<<"\n\nFile Not Found...\nProgram Terminated!";
            goto menu;
        }
        fin.seekg(0);
        while(fin.read((char*)&amt,sizeof(amt)))
        {
            int x=amt.item::retno();
            if(x==ino)
            {
                amt.pay();
                flag=1;
                break;
            }
        }
        if(flag==0)
            cout<<"\n\t\tItem does not exist....Please Retry!";
        getch();
        fin.close();
        goto menu;
    case 4:
        system("cls");
        gotoxy(20,20);
        cout<<"ARE YOU SURE, YOU WANT TO EXIT (Y/N)?";
        char yn;
        cin>>yn;
        if((yn=='Y')||(yn=='y'))
        {
            gotoxy(12,20);
            system("cls");
            cout<<"************************* THANKS
************************************";
            getch();
            exit(0);
        }
        else if((yn=='N')||(yn=='n'))
            goto menu;
        else
```

```
        {
            goto menu;
        }
    default:
        cout<<"\n\n\t\tWrong Choice....Please Retry!";
        getch();
        goto menu;
    }
    return 0;
}
```

OUTPUT:

```
                Super Market Billing
                ============================


          1.Bill Report

          2.Add/Remove/Edit Item

          3.Show Item Details

          4.Exit

          Please Enter Required Option: 2
```

```
                Bill Editor
                ================


          1.Add Item Details

          2.Edit Item Details

          3.Delete Item Details

          4.Back to Main Menu 1
```

```
                Bill Editor
                ================

          1.Add Item Details

          2.Edit Item Details

          3.Delete Item Details

          4.Back to Main Menu 1

    Item No: 2

    Name of the item: Cookies

    Manufacturing Date(dd-mm-yy): 22-03-22

    Price: 110

    Quantity: 25

    Tax percent: 9

    Discount percent: 5

          Item Added Successfully!
```

```
                Bill Editor
                ================

          1.Add Item Details

          2.Edit Item Details

          3.Delete Item Details

          4.Back to Main Menu 2

  Enter Item Number to be Edited :1
```

```
        Current Details are

Item No: 1

Name of the item: Fruits

Date : 27--7--22

Net amount: 4873.50

        Enter New Details


Item No: 1


Name of the item: Fruits


Manufacturing Date(dd-mm-yy): 25-07-22


Price: 95


Quantity: 50


Tax percent: 13


Discount percent: 5

        Item Details editted
```

```
                Bill Editor
                ===========


        1.Add Item Details

        2.Edit Item Details

        3.Delete Item Details

        4.Back to Main Menu 3


Enter Item Number to be deleted :2

        Item Succesfully Deleted
```

```
            Super Market Billing
            ============================


    1.Bill Report

    2.Add/Remove/Edit Item

    3.Show Item Details

    4.Exit

    Please Enter Required Option: 3
```

```
        Enter Item Number :1

Item No: 1

Name of the item: Fruits

Date : 27--7--22

Net amount: 4873.50


        *********************************************
                        DETAILS
        *********************************************

        PRICE                    :95.00

        QUANTITY                 :50.00
        TAX PERCENTAGE             :8.00
        DISCOUNT PERCENTAGE        :5.00


        NET AMOUNT             :Rs.4873.50
        *********************************************
```

```
            Super Market Billing
            ============================


        1.Bill Report

        2.Add/Remove/Edit Item

        3.Show Item Details

        4.Exit

        Please Enter Required Option: 1
```

```
            Bill Details
            ================

    1.All Items

    2.Back to Main menu

    Please Enter Required Option: 1
```

```
                        BILL DETAILS

ITEM NO   NAME      PRICE     QUANTITY   TAX %   DISCOUNT %  NET AMOUNT



1         Fruits    95.00     50.00      8.00    5.00        4873.50
1         Fruits    95.00     9.00       8.00    5.00        877.23
1         Fruits    97.00     50.00      10.00   5.00        5068.25


                  Grand Total=10818.98
```

```
                Super Market Billing
                ============================

          1.Bill Report

          2.Add/Remove/Edit Item

          3.Show Item Details

          4.Exit

          Please Enter Required Option: 4
```

```
   ARE YOU SURE, YOU WANT TO EXIT (Y/N)?y
```

```
************************* THANKS ************************************
PS D:\PPS-II\C-plus-plus-Program>
```