OOP AND COMPUTER GRAPHICS LABORATORY

Practical Examination
SPPU AY 2020-21
Semester :- 1

SUBMITTED BY :-

NAME :- OJUS PRAVIN JAISWAL
SEAT NO. :- S191094290

OOP Laboratory

Assignment No. 5

Problem Statement :- Write a function template for selection sort that inputs, sorts and outputs an integer array and a float array.

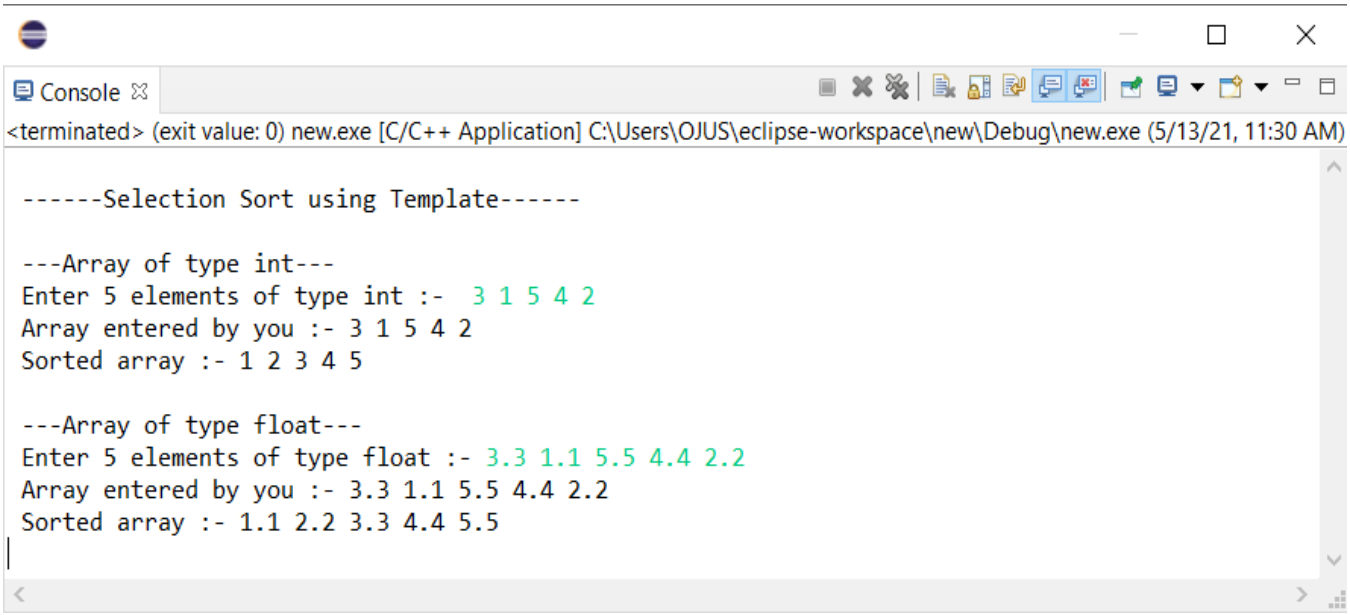
Solution :-

Program :

```
#include <iostream>
using namespace std;

template <typename X>
void Swap(X &a, X &b)
{
	X tem = a;
	a = b;
	b = tem;
}

template <typename T>
void selectionSort(T arr[5])
{
	int min;
	for (int i = 0; i < 5; i++)
	{
		min = i;
		for (int j = i + 1; j < 5; j++)
		{
			if (arr[min] > arr[j])
			{
				min = j;
			}
		}
		Swap(arr[min], arr[i]);
	}
	cout << "\n Sorted array :- ";
	for (int cnt = 0; cnt < 5; cnt++)
		cout << arr[cnt] << " ";
```

```
	cout << "\n";
}

int main()
{
	int intArr[5];
	cout << "\n ------Selection Sort using Template------\n";
	cout << "\n ---Array of type int---";
	cout << "\n Enter 5 elements of type int :- ";
	for (int cnt = 0; cnt < 5; cnt++)
		cin >> intArr[cnt];
	cout << " Array entered by you :- ";
	for (int cnt = 0; cnt < 5; cnt++)
		cout << intArr[cnt] << " ";
	selectionSort(intArr);
	cout << "\n ---Array of type float---";
	float floatArr[5];
	cout << "\n Enter 5 elements of type float :- ";
	for (int cnt = 0; cnt < 5; cnt++)
		cin >> floatArr[cnt];
	cout << " Array entered by you :- ";
	for (int cnt = 0; cnt < 5; cnt++)
		cout << floatArr[cnt] << " ";
	selectionSort(floatArr);
	return 0;
}
```

Output :



CG Laboratory

Assignment No. 6

Problem Statement :- Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL

i) Scaling

ii) Translation

iii) Rotation about an axis (X/Y/Z).

Solution :-

Program :

```
#include <math.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>

typedef float Matrix4x4 [4][4];
Matrix4x4 theMatrix;
float ptsIni[8][3]={{80,80,-100},{180,80,-100},{180,180,-100},{80,180,-
100},
                    {60,60,0},{160,60,0},{160,160,0},{60,160,0}};
//Realign above line while execution
// Initial Co-ordinates ofthe Cube to be Transformed
float ptsFin[8][3];
float refptX,refptY,refptZ;                     //Reference points
float TransDistX,TransDistY,TransDistZ;         //Translations along Axes
float ScaleX,ScaleY,ScaleZ;                     //Scaling Factors along Axes
float Alpha,Beta,Gamma,Theta;                   //Rotation angles about Axes
float A,B,C;                                    //Arbitrary Line Attributes
float aa,bb,cc;                                 //Arbitrary Line Attributes
float x1,y11,z1,x2,y2,z2;
int choice,choiceRot,choiceRef;
void matrixSetIdentity(Matrix4x4 m)     // Initialises the matrix as Unit
Matrix
{
 int i, j;
 for (i=0; i<4; i++)
 for (j=0; j<4; j++)
 m[i][j] = (i == j);
}

void matrixPreMultiply(Matrix4x4 a , Matrix4x4 b)
{// Multiplies matrix a times b, putting result in b
 int i,j;
 Matrix4x4 tmp;
 for (i = 0; i < 4; i++)
 for (j = 0; j < 4; j++)
 tmp[i][j]=a[i][0]*b[0][j]+a[i][1]*b[1][j]+a[i][2]*b[2][j]+a[i][3]*b[3][j];
 for (i = 0; i < 4; i++)
 for (j = 0; j < 4; j++)
 theMatrix[i][j] = tmp[i][j];
}
```

```
void Translate(int tx, int ty, int tz)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 m[0][3] = tx;
 m[1][3] = ty;
 m[2][3] = tz;
 matrixPreMultiply(m, theMatrix);
}
void Scale(float sx , float sy ,float sz)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 m[0][0] = sx;
 m[0][3] = (1 - sx)*refptX;
 m[1][1] = sy;
 m[1][3] = (1 - sy)*refptY;
 m[2][2] = sz;

 m[2][3] = (1 - sy)*refptZ;
 matrixPreMultiply(m , theMatrix);
}
void RotateX(float angle)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 angle = angle*22/1260;
 m[1][1] = cos(angle);
 m[1][2] = -sin(angle);
 m[2][1] = sin(angle);
 m[2][2] = cos(angle);
 matrixPreMultiply(m , theMatrix);
}
void RotateY(float angle)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 angle = angle*22/1260;
 m[0][0] = cos(angle);
 m[0][2] = sin(angle);
 m[2][0] = -sin(angle);
 m[2][2] = cos(angle);
 matrixPreMultiply(m , theMatrix);
}
void RotateZ(float angle)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 angle = angle*22/1260;
 m[0][0] = cos(angle);
 m[0][1] = -sin(angle);
 m[1][0] = sin(angle);
 m[1][1] = cos(angle);
 matrixPreMultiply(m , theMatrix);
}
```

```
void Reflect(void)
{
 Matrix4x4 m;
 matrixSetIdentity(m);
 switch(choiceRef)
 {
 case 1: m[2][2] = -1;
 break;
 case 2: m[0][0] = -1;
 break;
 case 3: m[1][1] = -1;
 break;
 }
 matrixPreMultiply(m , theMatrix);
}
void DrawRotLine(void)
{
 switch(choiceRot)
 {
 case 1: glBegin(GL_LINES);
 glVertex3s(-1000 ,B,C);
 glVertex3s( 1000 ,B,C);
 glEnd();
 break;
 case 2: glBegin(GL_LINES);
 glVertex3s(A ,-1000 ,C);
 glVertex3s(A ,1000 ,C);
 glEnd();
 break;
 case 3: glBegin(GL_LINES);
 glVertex3s(A ,B ,-1000);
 glVertex3s(A ,B ,1000);
 glEnd();
 break;
 case 4: glBegin(GL_LINES);
 glVertex3s(x1-aa*500 ,y11-bb*500 , z1-cc*500);
 glVertex3s(x2+aa*500 ,y2+bb*500 , z2+cc*500);
 glEnd();
 break;
 }
}
void TransformPoints(void)
{
 int i,k;
 float tmp ;
 for(k=0 ; k<8 ; k++)
 for (i=0 ; i<3 ; i++)
 ptsFin[k][i] = theMatrix[i][0]*ptsIni[k][0] + theMatrix[i][1]*ptsIni[k][1]
                + theMatrix[i][2]*ptsIni[k][2] + theMatrix[i][3];
// Realign above line while execution
}
void Axes(void)
{
 glColor3f (0.0, 0.0, 0.0);               // Set the color to BLACK
 glBegin(GL_LINES);                       // Plotting X-Axis
 glVertex2s(-1000 ,0);
 glVertex2s( 1000 ,0);
 glEnd();
 glBegin(GL_LINES);                       // Plotting Y-Axis
 glVertex2s(0 ,-1000);
 glVertex2s(0 , 1000);
 glEnd();
}
```

```
void Draw(float a[8][3])                  //Display the Figure
{
 int i;
 glColor3f (0.7, 0.4, 0.7);
 glBegin(GL_POLYGON);
 glVertex3f(a[0][0],a[0][1],a[0][2]);
 glVertex3f(a[1][0],a[1][1],a[1][2]);
 glVertex3f(a[2][0],a[2][1],a[2][2]);
 glVertex3f(a[3][0],a[3][1],a[3][2]);
 glEnd();
 i=0;
 glColor3f (0.8, 0.6, 0.5);
 glBegin(GL_POLYGON);
 glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);
 glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);
 glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);
 glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);
 glEnd();
 glColor3f (0.2, 0.4, 0.7);
 glBegin(GL_POLYGON);
 glVertex3f(a[0][0],a[0][1],a[0][2]);
 glVertex3f(a[3][0],a[3][1],a[3][2]);
 glVertex3f(a[7][0],a[7][1],a[7][2]);

 glVertex3f(a[4][0],a[4][1],a[4][2]);
 glEnd();
 i=1;
 glColor3f (0.5, 0.4, 0.3);
 glBegin(GL_POLYGON);
 glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);
 glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);
 glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);
 glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);
 glEnd();
 i=2;
 glColor3f (0.5, 0.6, 0.2);
 glBegin(GL_POLYGON);
 glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);
 glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);
 glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);
 glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);
 glEnd();
 i=4;
 glColor3f (0.7, 0.3, 0.4);
 glBegin(GL_POLYGON);
 glVertex3f(a[0+i][0],a[0+i][1],a[0+i][2]);
 glVertex3f(a[1+i][0],a[1+i][1],a[1+i][2]);
 glVertex3f(a[2+i][0],a[2+i][1],a[2+i][2]);
 glVertex3f(a[3+i][0],a[3+i][1],a[3+i][2]);
 glEnd();
}
```

```
void display(void)
{
 glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 Axes();
 glColor3f (1.0, 0.0, 0.0);                // Set the color to RED
 Draw(ptsIni);
 matrixSetIdentity(theMatrix);
 switch(choice)
 {
 case 1:   Translate(TransDistX , TransDistY ,TransDistZ);
 break;
 case 2:   Scale(ScaleX, ScaleY, ScaleZ);
 break;
 case 3:   switch(choiceRot)
 {
 case 1: DrawRotLine();
 Translate(0,-B,-C);
 RotateX(Alpha);
 Translate(0,B,C);
 break;
 case 2: DrawRotLine();
 Translate(-A,0,-C);
 RotateY(Beta);
 Translate(A,0,C);
 break;
 case 3: DrawRotLine();
 Translate(-A,-B,0);
 RotateZ(Gamma);
 Translate(A,B,0);
 break;
 case 4: DrawRotLine();
 float MOD =sqrt((x2-x1)*(x2-x1) + (y2-y11)*(y2-y11) + (z2-z1)*(z2-z1));
 aa = (x2-x1)/MOD;

 bb = (y2-y11)/MOD;
 cc = (z2-z1)/MOD;
 Translate(-x1,-y11,-z1);
 float ThetaDash;
 ThetaDash = 1260*atan(bb/cc)/22;
 RotateX(ThetaDash);
 RotateY(1260*asin(-aa)/22);
 RotateZ(Theta);
 RotateY(1260*asin(aa)/22);
 RotateX(-ThetaDash);
 Translate(x1,y11,z1);
 break;
 }
 break;
 case 4:    Reflect();
 break;
 }
 TransformPoints();
 Draw(ptsFin);
 glFlush();
}
```

```
void init(void)
 {
 glClearColor (1.0, 1.0, 1.0, 1.0);
     // Set the Background color to WHITE
 glOrtho(-454.0, 454.0, -250.0, 250.0, -250.0, 250.0);
     // Set the no. of Co-ordinates along X & Y axes and their gappings
 glEnable(GL_DEPTH_TEST);
     // To Render the surfaces Properly according to their depths
 }
int main (int argc, char **argv)
{
 glutInit(&argc, argv);
 glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
 glutInitWindowSize (1362, 750);
 glutInitWindowPosition (0, 0);
 glutCreateWindow (" Basic Transformations ");
 init ();
 printf("Enter your choice
number:\n1.Translation\n2.Scaling\n3.Rotation\n4.Reflection\n=>");
 scanf("%d",&choice);
 switch(choice)
 {
 case 1:printf("Enter Translation along X, Y & Z\n=>");
 scanf("%f%f%f",&TransDistX , &TransDistY , &TransDistZ);
 break;
 case 2:printf("Enter Scaling ratios along X, Y & Z\n=>");
 scanf("%f%f%f",&ScaleX , &ScaleY , &ScaleZ);
 break;
 case 3:printf("Enter your choice for Rotation about axis:\n1.parallel to
X-axis.(y=B & z=C)\n2.parallel to Y-axis.(x=A & z=C)\n3.parallel to Z-
axis.(x=A & y=B)\n4.Arbitrary line passing through (x1,y1,z1) &
(x2,y2,z2)/n =>");
//Realign above line while execution
 scanf("%d",&choiceRot);
 switch(choiceRot)
 {
 case 1: printf("Enter B & C: ");
 scanf("%f  %f",&B,&C);
 printf("Enter Rot. Angle Alpha: ");
 scanf("%f",&Alpha);


 break;
 case 2: printf("Enter A & C: ");
 scanf("%f %f",&A,&C);
 printf("Enter Rot. Angle Beta: ");
 scanf("%f",&Beta);
 break;
 case 3: printf("Enter A & B: ");
 scanf("%f  %f",&A,&B);
 printf("Enter Rot. Angle Gamma: ");
 scanf("%f",&Gamma);
 break;
 case 4: printf("Enter values of x1 ,y1 & z1:\n");
 scanf("%f %f %f",&x1,&y11,&z1);
 printf("Enter values of x2 ,y2 & z2:\n");
 scanf("%f %f %f",&x2,&y2,&z2);
 printf("Enter Rot. Angle Theta: ");
 scanf("%f",&Theta);
 break;
 }
```

```
 break;
 case 4:     printf("Enter your choice for reflection about plane:\n1.X-
Y\n2.Y-Z\n3.X-Z\n=>");
 scanf("%d",&choiceRef);
 break;
 default:    printf("Please enter a valid choice!!!\n");
 return 0;
 }
 glutDisplayFunc(display);
 glutMainLoop();
 return 0;
}
```

Output :

Enter your choice number:
1.Translation
2.Scaling
3.Rotation
4.Reflection
=>4
Enter your choice for reflection about plane:
1.X-Y
2.Y-Z
3.X-Z
=>3