Department of Computer Science & Engineering

Siddaganga Institute of Technology, Tumakuru-3

(An Autonomous Institute affiliated to VTU,Accredited by NBA)**2019-20**

TABLE OF CONTENTS

S. No.	Content	Page No.
1.	Problem Statement	1
2.	Implementation (Code)	2 - 7
3.	Output	8 - 9

Bubble Sort Simulation

Bubble sort which also referred sometimes as sinking sort is a sorting technique where the concepts lies around a lighter body bubbling up to the surface or heavier body sinking to the bottom, hence the name bubble sort or sinking sort. Here the heavier bodies are larger numbers and the lighter bodies are smaller numbers but as we the user giver the comparison the above may be vise versa. In bubble sort two adjacent elements are compared and a swap may occur if they are in the wrong order. As a result after each go in a list of elements one more element is placed at its correct position.

Below is the pesudocode for bubble sort:

```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    repeat
        swapped := false
        for i := 1 to n - 1 inclusive do
            if A[i - 1] > A[i] then
                swap(A[i - 1], A[i])
                swapped = true
            end if
        end for
        n := n - 1
    until not swapped
end procedure
```

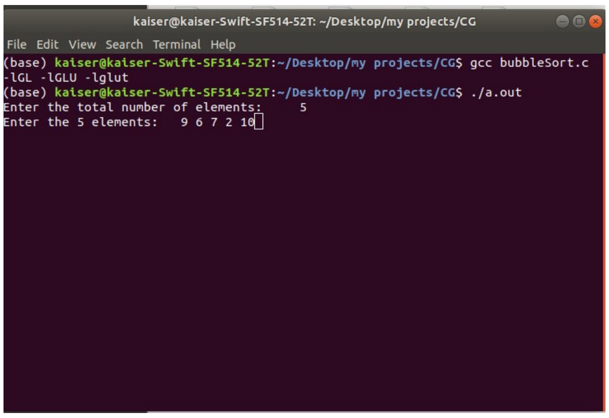
IMPLEMENTATION

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

#define WINDOW_SIZE 600

#define SIZE 7

#define y1 275
#define y2 325

#define arrow_Y 335
#define arrow_HEIGHT 20
#define array_BEGIN 100
#define array_WIDTH 50
#define array_GAP 10

int moveNext = 0;
int j_arrow_X = -10;
int j_plus_one_arrow_X = -10;

struct myDataType {
      int value;
      int x1, x2;
};

struct myDataType arr[SIZE];
int iT;
int iIndex;
int jIndex;

int startFlag = 1;
char process[] = "Please click on Start to begin the bubble sort";

void fnInitArray() {
      printf("Enter the total number of elements:\t");
      scanf("%d", &iT);
      while(iT > SIZE || iT <= 0) {
            printf("Number of elements can range from 1 to %d. Enter the total 
number of elements:\t", SIZE);
            scanf("%d", &iT);
      }

      iIndex = 0;
      jIndex = -2;
```

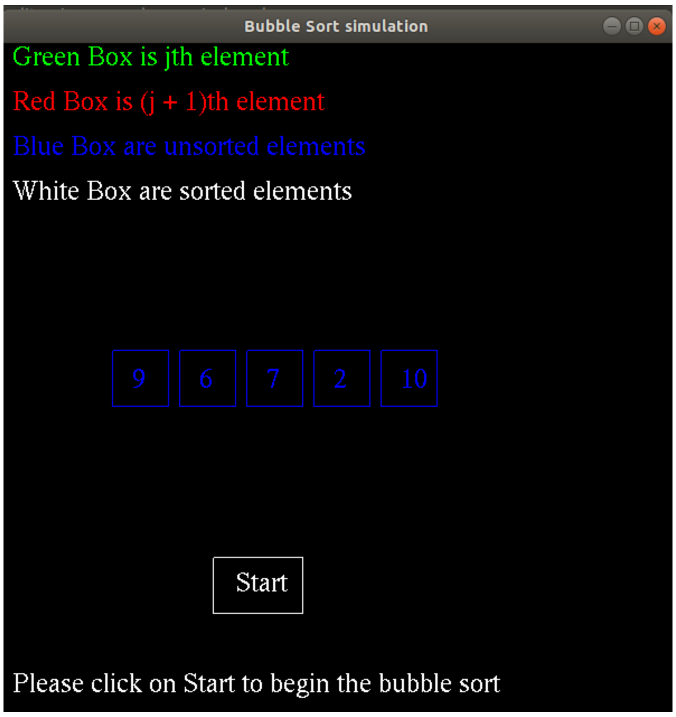
```
	printf("Enter the %d elements:\t", iT);

	for(int i = 0; i < iT; i++) {
		arr[i].x1 = array_BEGIN + i * (array_WIDTH + array_GAP);
		arr[i].x2 = arr[i].x1 + array_WIDTH;
		scanf("%d", &arr[i].value);
	}
}

void drawstring(float x, float y, const char *string)
{
	const char *c;
	glRasterPos2f(x, y);

	for (c = string; *c != '\0'; c++)
	{
		glColor3f(1, 1, 1);
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
	}
}

void fnDrawArray() {
	for(int i = 0; i < iT; i++) {
		glBegin(GL_LINE_LOOP);
		if(i >= iT - iIndex) {
			glColor3f(1, 1, 1);
		}
		else if(i == jIndex + 1) {
			glColor3f(1, 0, 0);
			j_plus_one_arrow_X = (arr[i].x1 + arr[i].x2) / 2;
		}
		else if(i == jIndex) {
			glColor3f(0, 1, 0);
			j_arrow_X = (arr[i].x1 + arr[i].x2) / 2;
		}
		else {
			glColor3f(0, 0, 1);
		}
		glVertex2d(arr[i].x1, y1);
		glVertex2d(arr[i].x1, y2);
		glVertex2d(arr[i].x2, y2);
		glVertex2d(arr[i].x2, y1);
		glEnd();
		char str[10];
		snprintf(str, sizeof(str), "%d", arr[i].value);
		drawstring(arr[i].x1 + 17, y1 + 17, str);
	}

}
```

```
struct button {
      int x11, y11, x22, y22;
      int state;
      char str[10];
};


struct button initButton(int x111, int y111, int x222, int y222, char *str1) {
      struct button btn;
      btn.x11 = x111;
      btn.y11 = y111;
      btn.x22 = x222;
      btn.y22 = y222;
      btn.state = 1;
      strcpy(btn.str, str1);

      return btn;
}

void draw(struct button btn) {

      glColor3f(1, 1, 1);

      drawstring(btn.x11 + 10, btn.y11 + 10, btn.str);

      glBegin(GL_LINE_LOOP);
      glVertex2d(btn.x11 - 10, btn.y11 - 10);
      glVertex2d(btn.x11 - 10, btn.y11 + 40);
      glVertex2d(btn.x11 + 70, btn.y11 + 40);
      glVertex2d(btn.x11 + 70, btn.y11 - 10);
      glEnd();
}

struct button togglestate(struct button btn) {
      btn.state = (btn.state == 1) ? 0 : 1;
      return btn;
}

int insidebutton(struct button btn, int x, int y) {
      if (x > btn.x11 && x < btn.x22 && y > btn.y11 && y < btn.y22)
            return 1;
      else return 0;
}

char Next[] = "Start";

struct button nextButton;

void mouse(int btn, int state, int x, int y)
{
      y = WINDOW_SIZE - y;

      if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
      {
            if (insidebutton(nextButton, x, y))
            {
                  if(startFlag) {
                        jIndex = 0;
                        startFlag = 0;
                        strcpy(nextButton.str,"Next");
                        glutPostRedisplay();
                        strcpy(process, "");
                        return;
                  }
                  if(!strcmp(nextButton.str, "End")) {exit(0);}
                  if(iIndex >= iT - 1) {
                        j_plus_one_arrow_X = - 10;
                        strcpy(nextButton.str,"End");
                        strcpy(process, "Successfully sorted");
                        glutPostRedisplay();
                        return;
                  }

                  strcpy(process, "");
                  if(arr[jIndex].value > arr[jIndex + 1].value) {
                        int temp = arr[jIndex].value;
                        arr[jIndex].value = arr[jIndex + 1].value;
                        arr[jIndex + 1].value = temp;
                        strcpy(process, "Swap occured");
                  }
                  glutPostRedisplay();

                  jIndex++;
                  if(jIndex >= iT - 1 - iIndex) {jIndex = 0; iIndex++;}
                  if(iIndex >= iT - 1) {
                        j_plus_one_arrow_X = - 10;
                        strcpy(nextButton.str,"End");
                        strcpy(process, "Successfully sorted");
                        glutPostRedisplay();
                        return;
                  }
                  nextButton = togglestate(nextButton);
            }

      }

      if (btn == GLUT_LEFT_BUTTON && state == GLUT_UP)
      {
            if (insidebutton(nextButton, x, y))
            {
                  nextButton = togglestate(nextButton);
            }
      }

}

void fnColorIndex() {
      glColor3f(0, 1, 0);
      drawstring(10, WINDOW_SIZE - 20, "Green Box is jth element");
      glColor3f(1, 0, 0);
      drawstring(10, WINDOW_SIZE - 60, "Red Box is (j + 1)th element");
      glColor3f(0, 0, 1);
      drawstring(10, WINDOW_SIZE - 100, "Blue Box are unsorted elements");
      glColor3f(1, 1, 1);
      drawstring(10, WINDOW_SIZE - 140, "White Box are sorted elements");
}

void fnProcess() {
      glColor3f(1, 1, 1);
      drawstring(10,20, process);
}

void fnArrow() {
      glBegin(GL_LINE_STRIP);
      glColor3f(0, 1, 0);
      glVertex2d(j_arrow_X, arrow_Y + arrow_HEIGHT);
      glVertex2d(j_arrow_X, arrow_Y);
      glVertex2d(j_arrow_X - 5, arrow_Y + 5);
      glVertex2d(j_arrow_X, arrow_Y);
      glVertex2d(j_arrow_X + 5, arrow_Y + 5);
      glEnd();

      glBegin(GL_LINE_STRIP);
      glColor3f(1, 0, 0);
      glVertex2d(j_plus_one_arrow_X, arrow_Y + arrow_HEIGHT);
      glVertex2d(j_plus_one_arrow_X, arrow_Y);
      glVertex2d(j_plus_one_arrow_X - 5, arrow_Y + 5);
      glVertex2d(j_plus_one_arrow_X, arrow_Y);
      glVertex2d(j_plus_one_arrow_X + 5, arrow_Y + 5);
      glEnd();
}

void display()
{
      glClearColor(0, 0, 0, 0);
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      draw(nextButton);
      fnDrawArray();
      fnColorIndex();
      fnProcess();
      fnArrow();
      glFlush();
      glutSwapBuffers();
}

int main(int argc, char **argv)
{
      fnInitArray();

      nextButton = initButton(200, 100, 275, 150, Next);

      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(WINDOW_SIZE, WINDOW_SIZE);
      glutCreateWindow("Bubble Sort simulation");
      glutDisplayFunc(display);
      glutMouseFunc(mouse);
      glutPostRedisplay();
      glEnable(GL_DEPTH_TEST);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0, WINDOW_SIZE, 0, WINDOW_SIZE);
      glMatrixMode(GL_MODELVIEW);
      glEnable(GL_COLOR_MATERIAL);
      glutMainLoop();

      return 0;
}
```

OUTPUT


kaiser@kaiser-Swift-SF514-52T: ~/Desktop/my projects/CG
File Edit View Search Terminal Help

```
(base) kaiser@kaiser-Swift-SF514-52T:~/Desktop/my projects/CG$ gcc bubbleSort.c
-lGL -lGLU -lglut
(base) kaiser@kaiser-Swift-SF514-52T:~/Desktop/my projects/CG$ ./a.out
Enter the total number of elements:     5
Enter the 5 elements:   9 6 7 2 10
```



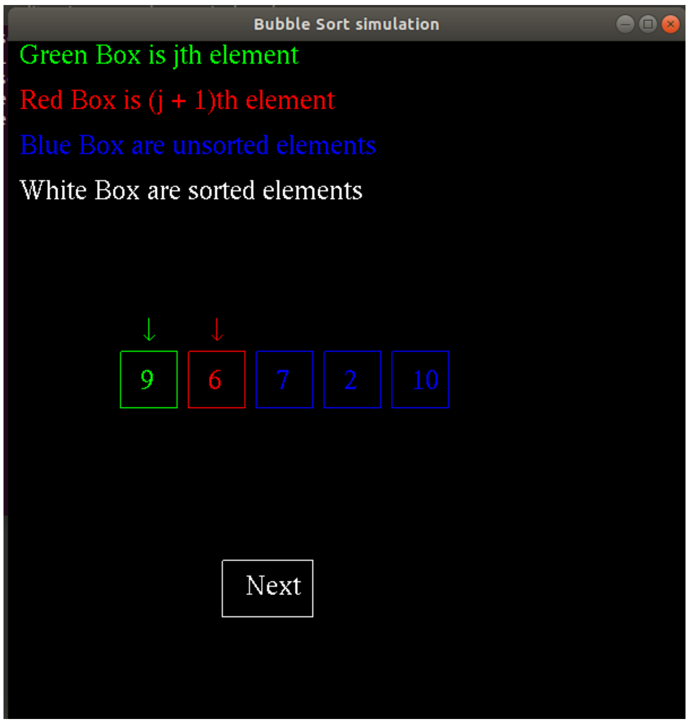
Green Box is jth element

Red Box is (j + 1)th element

Blue Box are unsorted elements

White Box are sorted elements

Please click on Start to begin the bubble sort

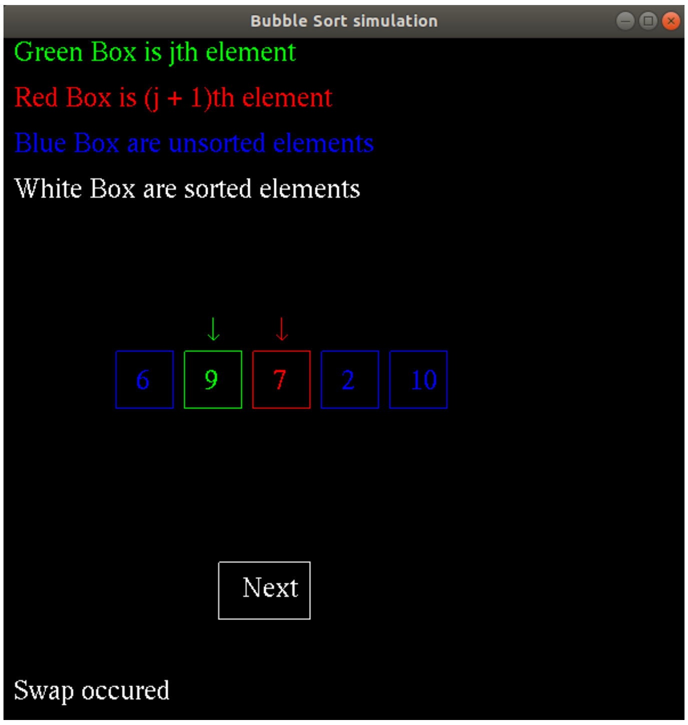


Green Box is jth element

Red Box is (j + 1)th element

Blue Box are unsorted elements

White Box are sorted elements



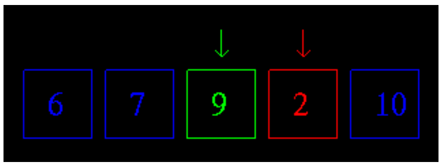
Green Box is jth element

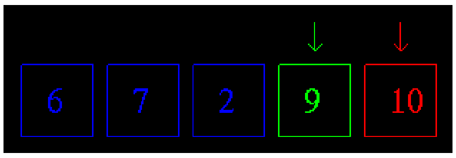
Red Box is (j + 1)th element

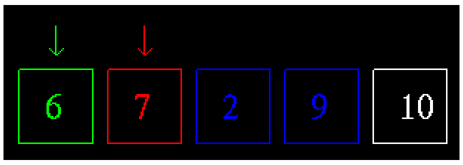
Blue Box are unsorted elements

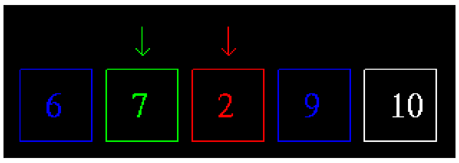
White Box are sorted elements

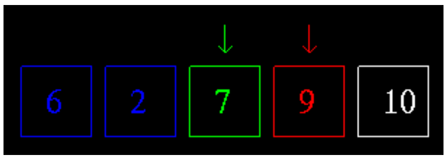
Swap occured

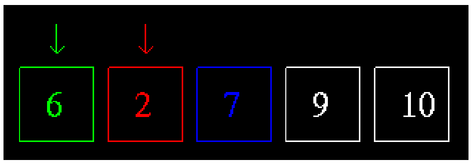


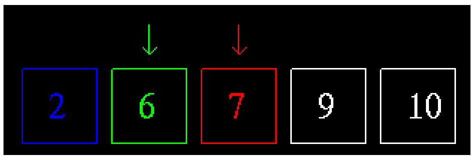


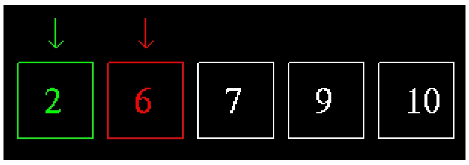


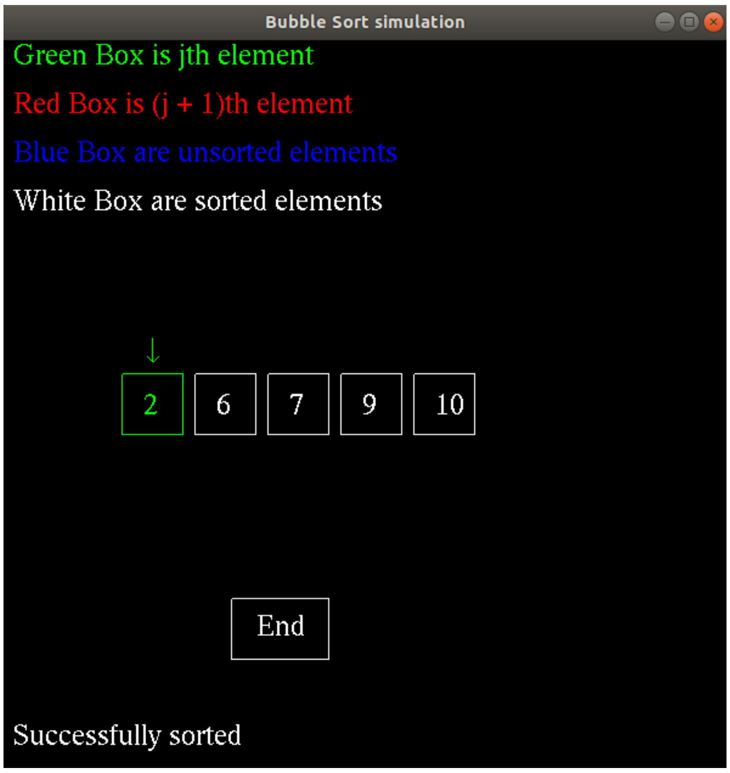












Green Box is jth element

Red Box is (j + 1)th element

Blue Box are unsorted elements

White Box are sorted elements

Successfully sorted