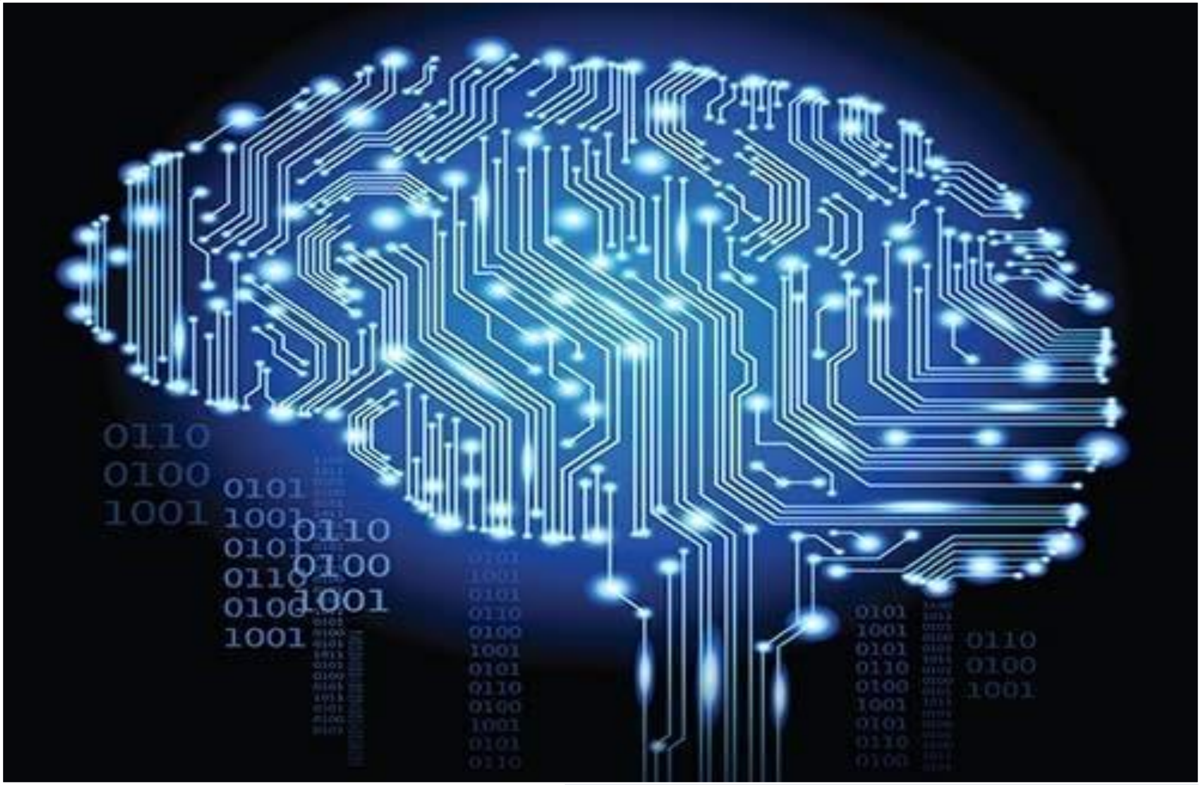
Submitted To:
Sheetal Singh

Design and Analysis of Algorithms



Submitted By:

Karan Sharma

Roll Number: 20201456

Course : BSC Computer Science Hons.

(Semester 4th)

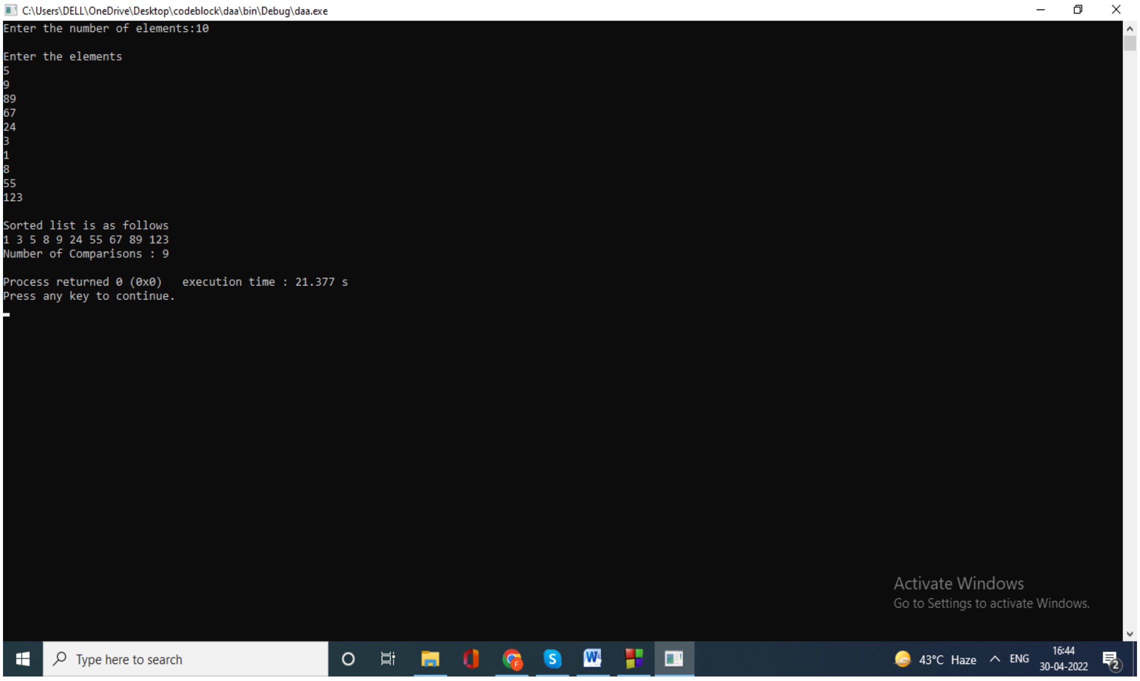
1. **a) Implement Insertion Sort (The program should report the number of comparisons)**

   Solution:

```
#include<iostream>

using namespace std;

int main()
{
   int i,j,n,temp,a[30], Comp=0;
   cout<<"Enter the number of elements:";
   cin>>n;
   cout<<"\nEnter the elements\n";

   for(i=0;i<n;i++)
   {
      cin>>a[i];
   }

   for(i=1;i<=n-1;i++)
   {
      temp=a[i];
      j=i-1;

      while((temp<a[j])&&(j>=0))
      {
         a[j+1]=a[j];
         j=j-1;
      }

      a[j+1]=temp;
      Comp++;
   }
```

```
    cout<<"\nSorted list is as follows\n";
    for(i=0;i<n;i++)
    {
       cout<<a[i]<<" ";
    }
     cout<<"\nNumber of Comparisons : "<<Comp<<endl;
    return 0;
}
```

Output:



C:\Users\DELL\OneDrive\Desktop\codeblock\daa\bin\Debug\daa.exe

```
Enter the number of elements:10

Enter the elements
5
9
89
67
24
3
1
8
55
123

Sorted list is as follows
1 3 5 8 9 24 55 67 89 123
Number of Comparisons : 9

Process returned 0 (0x0)   execution time : 21.377 s
Press any key to continue.
```

b) Implement Merge Sort(The program should report the number of comparisons)

Solution:

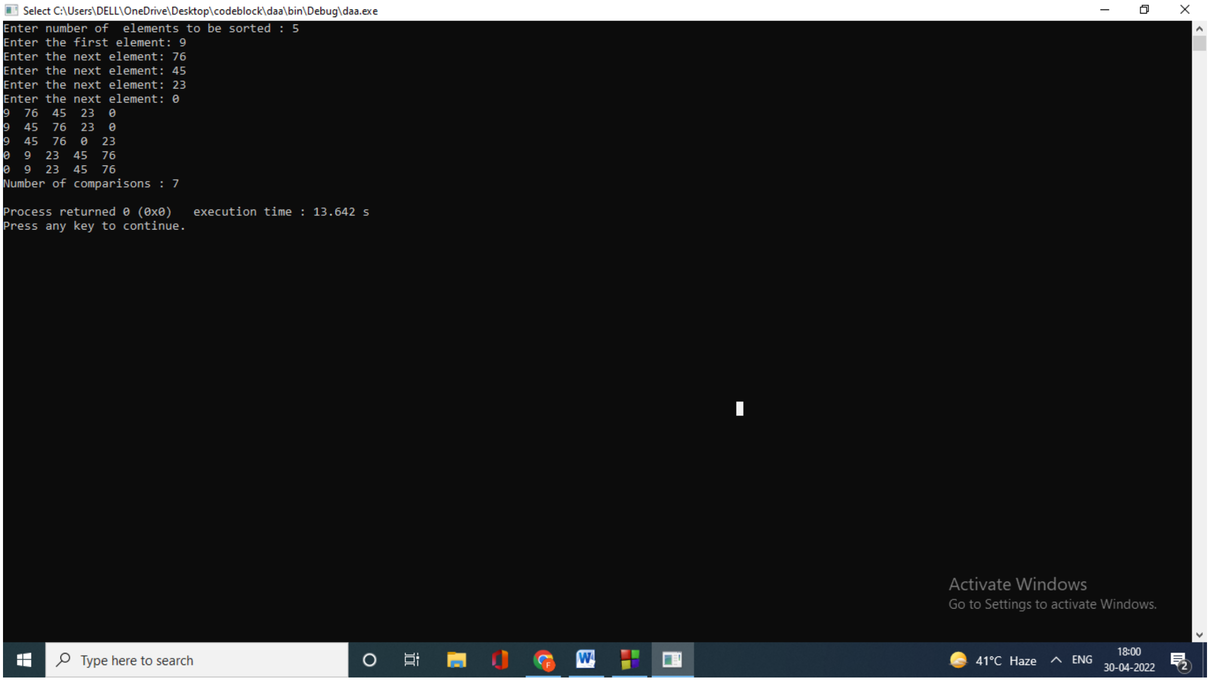
```
#include<iostream>

#include<stdio.h>

using namespace std;
```

```
        int count = 0;

        int n = 0;

        const int MAX_ITEMS = 100;

        void merge(int values[], int leftFirst, int leftLast, int rightFirst, int
rightLast);

        void printarray( int a[], int n);

        void mergesort(int a[], int start, int end){


       if(start < end){

           int mid = (start+end)/2;

           mergesort(a,start, mid);

           mergesort(a,mid+1,end);

            merge(a, start,mid, mid+1, end);

        }

  }

 void merge(int values[], int leftFirst, int leftLast, int rightFirst, int rightLast){

      int temparray[MAX_ITEMS];

      int index = leftFirst;

      int saveFirst = leftFirst;


      while((leftFirst <= leftLast)  && ( rightFirst <= rightLast)){


         if(values[leftFirst] < values[rightFirst]){

            temparray[index]  = values[leftFirst];
            leftFirst++;
         }
         else
         {
            temparray[index]  = values[rightFirst];
            rightFirst++;
         }
         index++;
         count++;
      }

      while(leftFirst <= leftLast){

         temparray[index] = values[leftFirst];
         leftFirst++;
         index++;

      }
      while(rightFirst <= rightLast){
         temparray[index] = values[rightFirst];
         rightFirst++;
         index++;
```

```
      }


      for(index = saveFirst; index <= rightLast; index++)
         values[index] = temparray[index];
      printarray(values,n);
      cout << endl;


   }


void printarray( int a[], int n){
   for (int i=0; i < n; i++)
      cout << a[i] << "  ";
}


int main(){


   cout << "Enter number of  elements to be sorted : ";
   cin >>n;


   int a[MAX_ITEMS];
   for (int i=0; i < n; i++){
      if(i==0)
         cout << "Enter the first element: ";
```

```
        else

           cout << "Enter the next element: ";

        cin >>     a[i];

    }


    int start = 0;

    int end = n-1;

   mergesort(a, start, end);

    printarray(a, n);

    cout << endl;

    cout  << "Number of comparisons : "<< count << endl;

return 0;

}
```

Output:



Select C:\Users\DELL\OneDrive\Desktop\codeblock\daa\bin\Debug\daa.exe

```
Enter number of  elements to be sorted : 5
Enter the first element: 9
Enter the next element: 76
Enter the next element: 45
Enter the next element: 23
Enter the next element: 0
9  76  45  23  0
9  45  76  23  0
9  45  76  0  23
0  9  23  45  76
0  9  23  45  76
Number of comparisons : 7

Process returned 0 (0x0)   execution time : 13.642 s
Press any key to continue.
```

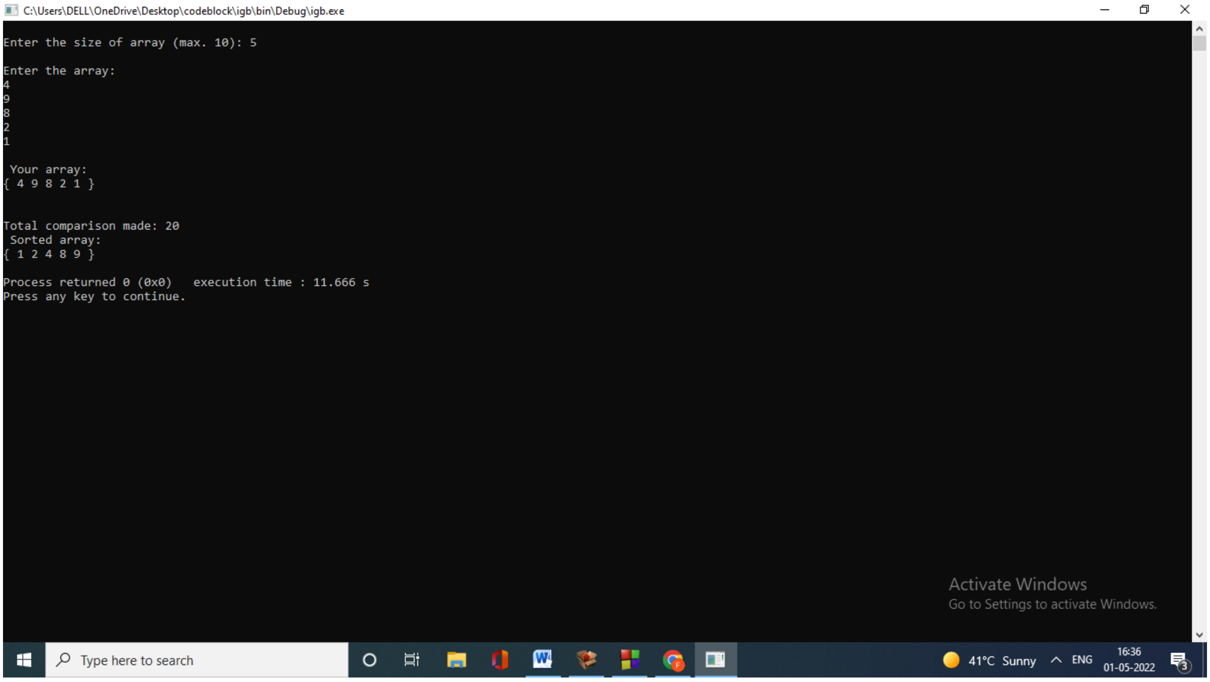
2 Implement Heap Sort (The program should report the number of comparisons)

Solution:

```
#include<iostream>
#include<conio.h>

using namespace std;

int comparison = 0;

void display(int *a, int size) {
  cout<<"{ ";
  for(int i=0; i<size; i++ )
      cout<<a[i]<<' ';
  cout<<"}"<<endl;
}

void swap(int *a, int x, int y){
 int temp = a[y];
 a[y] = a[x];
 a[x] = temp;
}

void maxHeapify(int *a, int index, int heapSize)
{
 int left = index*2 + 1;
 int right = index*2 + 2;
 int largest = index;

 if(left < heapSize && a[left] > a[largest]){
  largest = left;
       comparison+=2;
 }
```

```
  if(right < heapSize && a[right] > a[largest]){
    largest = right;
            comparison++;
  }

  if(largest != index){
    comparison++;
    swap(a, largest, index);
    maxHeapify(a, largest, heapSize);
  }
}

void buildMaxHeap(int *a, int n)
{
  for (int i = (n/2) - 1; i >= 0; i--) {
    maxHeapify(a, i, n);
    comparison++;
  }
}

void heapSort(int *a, int size)
{
  buildMaxHeap(a, size);
  int heapSize = size, i;
  for(i=size-1; i>=0; i--) {
          swap(a, 0, i);
          heapSize--;
          comparison++;
          maxHeapify(a,0,heapSize);
  }
}

int main()
{
 int size, i, *arr;
 cout<<"\nEnter the size of array (max. 10): ";
 cin>>size;
 arr = new int[size];

 cout<<"\nEnter the array: \n";
 for(i=0; i<size; i++)
   cin>>arr[i];


 cout<<"\n Your array: \n";
 display(arr, size);
 getch();


 heapSort(arr, size);
 cout<<"\n\nTotal comparison made: "<<comparison;

 cout<<"\n Sorted array: \n";
 display(arr, size);
 getch();

 return 0;
}
```

Output:



C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
41°C Sunny
ENG
16:36
01-05-2022

```
Enter the size of array (max. 10): 5

Enter the array:
4
9
8
2
1

 Your array:
{ 4 9 8 2 1 }


Total comparison made: 20
 Sorted array:
{ 1 2 4 8 9 }

Process returned 0 (0x0)   execution time : 11.666 s
Press any key to continue.
```

3. Implement Randomized Quick sort (The program should report the number of comparisons)

Solution:

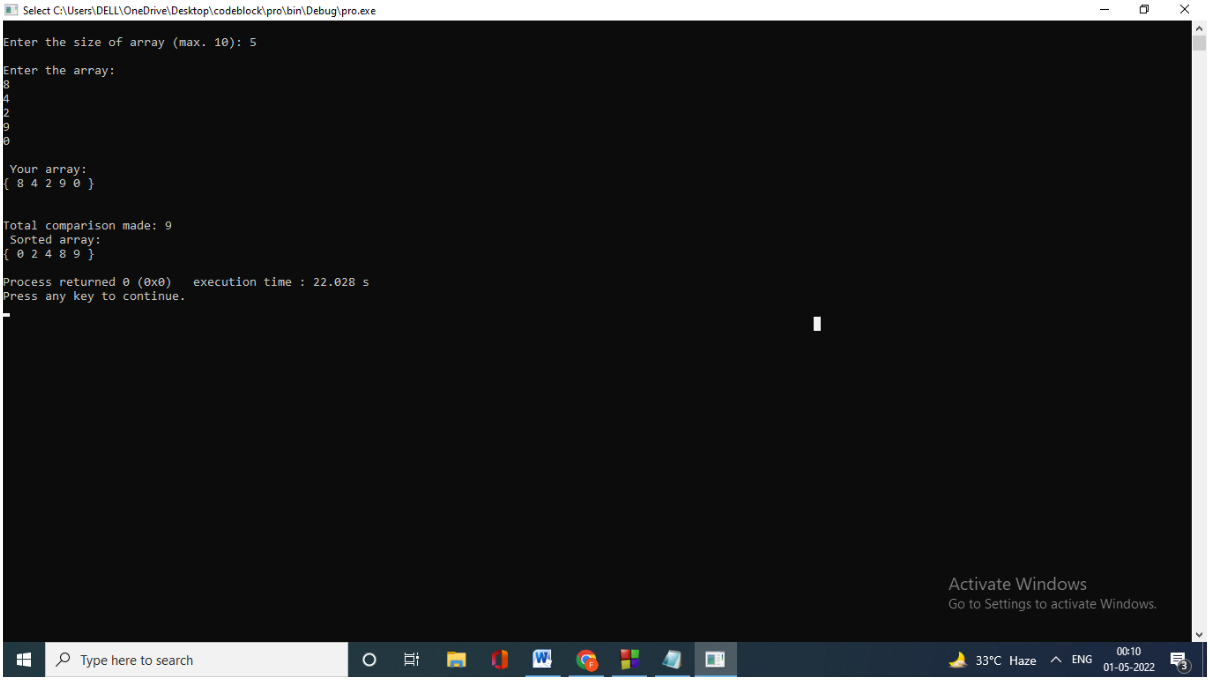
```
 #include<conio.h>

#include<iostream>

#include<stdlib.h>

#include<stdio.h>


using namespace std;


int comparison = 0;


void display(int *a, int size) {

    cout<<"{ ";
```

```
  for(int i=0; i<size; i++ )
      cout<<a[i]<<' ';
  cout<<"}"<<endl;
}

void swap(int *a, int x, int y){
 int temp = a[y];
 a[y] = a[x];
 a[x] = temp;
}

int partition(int *a, int p, int r)
{
 int i = p-1, j, x;
 for (j = p; j<r; j++)
  if(a[j] <= a[r]){
   comparison+=2;
   i++;
   swap(a, j, i);
  }
 swap(a, i+1, r);

 return i+1;
```

```
}

int randomizedPartition(int *a, int beg, int end)
{
  int t = (rand()%(end-beg)) + beg;
  swap(a, end, t);
  return partition(a, beg, end);
}

void randomizedQuickSort(int *a, int p, int r)
{
  if (p<r) {
    comparison++;
    int q = randomizedPartition(a, p, r);
    randomizedQuickSort(a, p, q-1);
    randomizedQuickSort(a, q+1, r);
  }
}

int main()
{

 int size, i, *arr;
```

```
}

 cout<<"\nEnter the size of array (max. 10): ";
 cin>>size;
 arr = new int[size];

 cout<<"\nEnter the array: \n";
 for(i=0; i<size; i++)
   cin>>arr[i];



 cout<<"\n Your array: \n";
 display(arr, size);
 getch();



 randomizedQuickSort(arr, 0, size-1);
 cout<<"\n\nTotal comparison made: "<<comparison;

 cout<<"\n Sorted array: \n";
 display(arr, size);
 getch();
 return 0;
}
```

Output:



Select C:\Users\DELL\OneDrive\Desktop\codeblock\pro\bin\Debug\pro.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
33°C Haze
ENG
00:10
01-05-2022

```
Enter the size of array (max. 10): 5

Enter the array:
8
4
2
9
0

 Your array:
{ 8 4 2 9 0 }


Total comparison made: 9
 Sorted array:
{ 0 2 4 8 9 }

Process returned 0 (0x0)   execution time : 22.028 s
Press any key to continue.
```

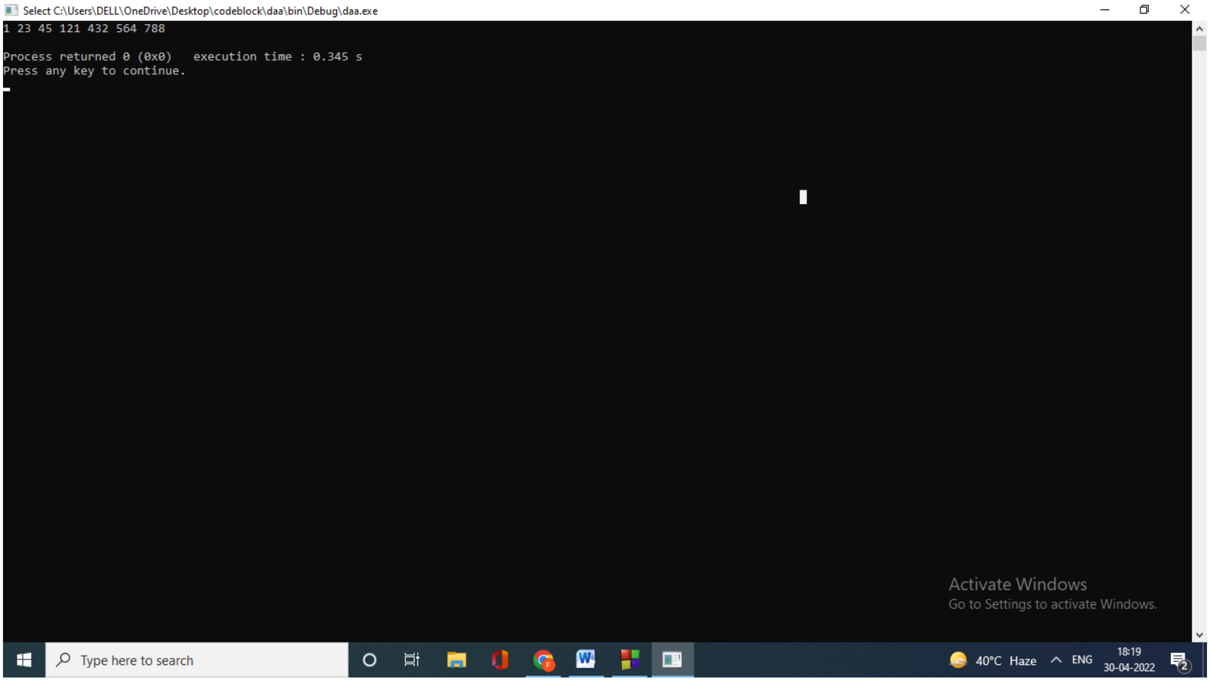
4. Implement Radix Sort

Solution:

```
#include <iostream>

using namespace std;

 int getMax(int array[], int n) {

      int max = array[0];

      for (int i = 1; i < n; i++)

          if (array[i] > max)

              max = array[i];

        return max;

      }

void countingSort(int array[], int size, int place) {

 const int max = 10;
```

```
  int output[size];
  int count[max];

  for (int i = 0; i < max; ++i)
    count[i] = 0;

  for (int i = 0; i < size; i++)
    count[(array[i] / place) % 10]++;

  for (int i = 1; i < max; i++)
    count[i] += count[i - 1];

  for (int i = size - 1; i >= 0; i--) {
    output[count[(array[i] / place) % 10] - 1] = array[i];
    count[(array[i] / place) % 10]--;
  }

  for (int i = 0; i < size; i++)
    array[i] = output[i];
}

void radixsort(int array[], int size) {
  int max = getMax(array, size);
  for (int place = 1; max / place > 0; place *= 10)
    countingSort(array, size, place);
}

void printArray(int array[], int size) {
  int i;
  for (i = 0; i < size; i++)
    cout << array[i] << " ";
  cout << endl;
}
int main() {
  int array[] = {121, 432, 564, 23, 1, 45, 788};
  int n = sizeof(array) / sizeof(array[0]);
  radixsort(array, n);
  printArray(array, n);

  return 0;
}
```

Output:


Select C:\Users\DELL\OneDrive\Desktop\codeblock\daa\bin\Debug\daa.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
40°C Haze
ENG
18:19
30-04-2022

```
1 23 45 121 432 564 788

Process returned 0 (0x0)   execution time : 0.345 s
Press any key to continue.
```

5. Create a Red-Black Tree and perform following operations on it:

i. Insert a node

ii. Delete a node

iii. Search for a number & also report the color of the node containing this number.

Solution:

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


 struct Node {
      int data;
    Node *parent;
    Node *left;
    Node *right;
```

```
        int color;
};

typedef Node *NodePtr;

class RedBlackTree {
 private:
NodePtr root;
NodePtr TNULL;

void initializeNULLNode(NodePtr node, NodePtr parent) {
  node->data = 0;
  node->parent = parent;
  node->left = nullptr;
  node->right = nullptr;
  node->color = 0;
}



void preOrderHelper(NodePtr node) {
  if (node != TNULL) {
    cout << node->data << " ";
    preOrderHelper(node->left);
    preOrderHelper(node->right);
  }
}



void inOrderHelper(NodePtr node) {
  if (node != TNULL) {
    inOrderHelper(node->left);
    cout << node->data << " ";
    inOrderHelper(node->right);
  }
}



void postOrderHelper(NodePtr node) {
  if (node != TNULL) {
    postOrderHelper(node->left);
    postOrderHelper(node->right);
    cout << node->data << " ";
  }
}

NodePtr searchTreeHelper(NodePtr node, int key) {
 if (node == TNULL || key == node->data) {
  return node;
 }

 if (key < node->data) {
  return searchTreeHelper(node->left, key);
 }
 return searchTreeHelper(node->right, key);
}


void deleteFix(NodePtr x) {
 NodePtr s;
 while (x != root && x->color == 0) {
  if (x == x->parent->left) {
   s = x->parent->right;
   if (s->color == 1) {
    s->color = 0;
    x->parent->color = 1;
    leftRotate(x->parent);
    s = x->parent->right;
   }
```

```
      if (s->left->color == 0 && s->right->color == 0) {
        s->color = 1;
        x = x->parent;
      } else {
        if (s->right->color == 0) {
          s->left->color = 0;
          s->color = 1;
          rightRotate(s);
          s = x->parent->right;
        }

        s->color = x->parent->color;
        x->parent->color = 0;
        s->right->color = 0;
        leftRotate(x->parent);
        x = root;
      }
    } else {
      s = x->parent->left;
      if (s->color == 1) {
        s->color = 0;
        x->parent->color = 1;
        rightRotate(x->parent);
        s = x->parent->left;
      }

      if (s->right->color == 0 && s->right->color == 0) {
        s->color = 1;
        x = x->parent;
      } else {
        if (s->left->color == 0) {
          s->right->color = 0;
          s->color = 1;
          leftRotate(s);
          s = x->parent->left;
        }

        s->color = x->parent->color;
        x->parent->color = 0;
        s->left->color = 0;
        rightRotate(x->parent);
        x = root;
      }
    }
  }
  x->color = 0;
```

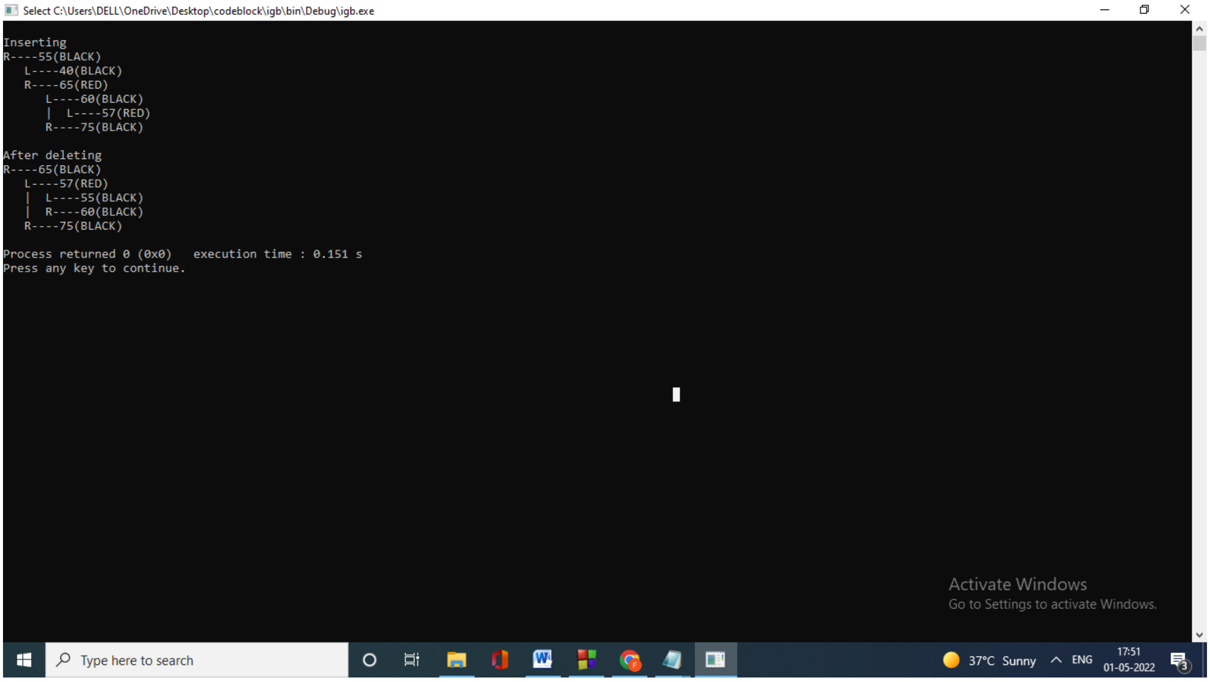
```
}

void rbTransplant(NodePtr u, NodePtr v) {
  if (u->parent == nullptr) {
    root = v;
  } else if (u == u->parent->left) {
    u->parent->left = v;
  } else {
    u->parent->right = v;
  }
  v->parent = u->parent;
}

void deleteNodeHelper(NodePtr node, int key) {
  NodePtr z = TNULL;
  NodePtr x, y;
  while (node != TNULL) {
    if (node->data == key) {
      z = node;
    }

    if (node->data <= key) {
      node = node->right;
  } else {
    node = node->left;
  }
}

if (z == TNULL) {
  cout << "Key not found in the tree" << endl;
  return;
}

y = z;
int y_original_color = y->color;
if (z->left == TNULL) {
  x = z->right;
  rbTransplant(z, z->right);
} else if (z->right == TNULL) {
  x = z->left;
  rbTransplant(z, z->left);
} else {
  y = minimum(z->right);
  y_original_color = y->color;
  x = y->right;
  if (y->parent == z) {
      x->parent = y;
    } else {
      rbTransplant(y, y->right);
      y->right = z->right;
      y->right->parent = y;
    }

    rbTransplant(z, y);
    y->left = z->left;
    y->left->parent = y;
    y->color = z->color;
  }
  delete z;
  if (y_original_color == 0) {
    deleteFix(x);
  }
}

void insertFix(NodePtr k) {
  NodePtr u;
  while (k->parent->color == 1) {
    if (k->parent == k->parent->parent->right) {
      u = k->parent->parent->left;
    if (u->color == 1) {
      u->color = 0;
      k->parent->color = 0;
      k->parent->parent->color = 1;
      k = k->parent->parent;
    } else {
      if (k == k->parent->left) {
        k = k->parent;
        rightRotate(k);
      }
      k->parent->color = 0;
      k->parent->parent->color = 1;
      leftRotate(k->parent->parent);
    }
  } else {
    u = k->parent->parent->right;

    if (u->color == 1) {
      u->color = 0;
      k->parent->color = 0;
      k->parent->parent->color = 1;
      k = k->parent->parent;
    } else {
        if (k == k->parent->right) {
          k = k->parent;
          leftRotate(k);
        }
        k->parent->color = 0;
        k->parent->parent->color = 1;
        rightRotate(k->parent->parent);
      }
    }
    if (k == root) {
      break;
    }
  }
  root->color = 0;
}

void printHelper(NodePtr root, string indent, bool last) {
  if (root != TNULL) {
    cout << indent;
    if (last) {
      cout << "R----";
      indent += "   ";
    } else {
```

```
   cout << "L----";
   indent += "|  ";
  }

  string sColor = root->color ? "RED" : "BLACK";
  cout << root->data << "(" << sColor << ")" << endl;
  printHelper(root->left, indent, false);
  printHelper(root->right, indent, true);
 }
}

 public:
RedBlackTree() {
 TNULL = new Node;
 TNULL->color = 0;
 TNULL->left = nullptr;
 TNULL->right = nullptr;
 root = TNULL;
}

void preorder() {
 preOrderHelper(this->root);
}
```

```
void inorder() {
  inOrderHelper(this->root);
}

void postorder() {
  postOrderHelper(this->root);
}

NodePtr searchTree(int k) {
  return searchTreeHelper(this->root, k);
}

NodePtr minimum(NodePtr node) {
  while (node->left != TNULL) {
    node = node->left;
  }
  return node;
}

NodePtr maximum(NodePtr node) {
  while (node->right != TNULL) {
    node = node->right;
  }
  return node;
}

NodePtr successor(NodePtr x) {
  if (x->right != TNULL) {
    return minimum(x->right);
  }

  NodePtr y = x->parent;
  while (y != TNULL && x == y->right) {
    x = y;
    y = y->parent;
  }
  return y;
}

NodePtr predecessor(NodePtr x) {
  if (x->left != TNULL) {
    return maximum(x->left);
  }

  NodePtr y = x->parent;
  while (y != TNULL && x == y->left) {
    x = y;
    y = y->parent;
  }

  return y;
}

void leftRotate(NodePtr x) {
  NodePtr y = x->right;
  x->right = y->left;
  if (y->left != TNULL) {
    y->left->parent = x;
  }
  y->parent = x->parent;
  if (x->parent == nullptr) {
    this->root = y;
  } else if (x == x->parent->left) {
    x->parent->left = y;
  } else {
    x->parent->right = y;
  }
  y->left = x;
  x->parent = y;
}

void rightRotate(NodePtr x) {
  NodePtr y = x->left;
  x->left = y->right;
  if (y->right != TNULL) {
    y->right->parent = x;
  }
  y->parent = x->parent;
  if (x->parent == nullptr) {
    this->root = y;
  } else if (x == x->parent->right) {
    x->parent->right = y;
  } else {
    x->parent->left = y;
  }
  y->right = x;
  x->parent = y;
}


void insert(int key) {
```

```
NodePtr node = new Node;
node->parent = nullptr;
node->data = key;
node->left = TNULL;
node->right = TNULL;
node->color = 1;

NodePtr y = nullptr;
NodePtr x = this->root;

while (x != TNULL) {
  y = x;
  if (node->data < x->data) {
    x = x->left;
  } else {
    x = x->right;
  }
}

node->parent = y;
if (y == nullptr) {
  root = node;
} else if (node->data < y->data) {
    y->left = node;
  } else {
    y->right = node;
  }

  if (node->parent == nullptr) {
    node->color = 0;
    return;
  }

  if (node->parent->parent == nullptr) {
    return;
  }

  insertFix(node);
}

NodePtr getRoot() {
  return this->root;
}

void deleteNode(int data) {
  deleteNodeHelper(this->root, data);
  }

  void printTree() {
    if (root) {
      printHelper(this->root, "", true);
    }
  }
};

int main() {
  RedBlackTree bst;
  cout << endl
     << "Inserting" << endl;
  bst.insert(55);
  bst.insert(40);
  bst.insert(65);
  bst.insert(60);
  bst.insert(75);
  bst.insert(57);

  bst.printTree();
  cout << endl
     << "After deleting" << endl;
  bst.deleteNode(40);

  bst.printTree();


  return 0;


}
```

Output:



Select C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe

```
Inserting
R----55(BLACK)
   L----40(BLACK)
   R----65(RED)
      L----60(BLACK)
      |  L----57(RED)
      R----75(BLACK)

After deleting
R----65(BLACK)
   L----57(RED)
   |  L----55(BLACK)
   |  R----60(BLACK)
   R----75(BLACK)

Process returned 0 (0x0)   execution time : 0.151 s
Press any key to continue.
```

6. Write a program to determine the LCS of two given sequences

Solution:

```
#include<iostream>

#include<conio.h>
```

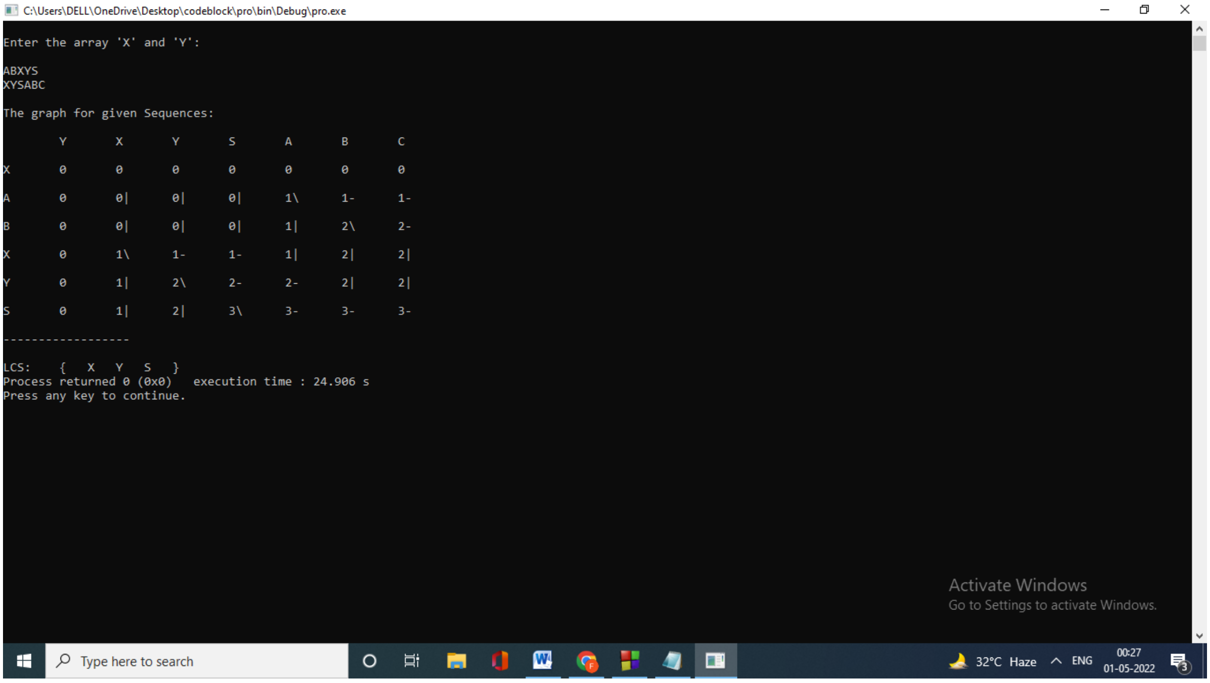
```
#include<stdio.h>

#include<string.h>

using namespace std;


  char b[20][20];

  char c[20][20];

  char *seq;

  int index = 0;




void fillSeq(char *y, int m, int n)

{

   while(b[m][n] != ' ')

   {

     switch(b[m][n])

     {

       case '\\':  seq[index++] = y[n-1];

                   m--;

                   n--;

                   break;


       case '|':   m--;

                   break;
```

```
      case '-':   n--;
                     break;

      default:    break;
    }
  }
}

void printLCS(char *x, char *y, int row, int col)
{
  cout<<"------------------\n\nLCS:\t"
      <<"{   ";
  fillSeq(y, row-1, col-1);
  for (int i = index-1; i > -1; i--) {
    cout<<seq[i]<<"   ";
  }
  cout<<"}";
}

void printTable(char *x, char *y, int row, int col)
{
  cout<<"\nThe graph for given Sequences:\n\n";


  cout<<"\tY\t";
  for(int i=0; i<col; i++)
    cout<<y[i]<<"\t";
  cout<<endl<<endl;


  for (int i = 0; i < row; i++) {
    if(i==0)
      cout<<"X\t";
    else
      cout<<x[i-1]<<"\t";


    for (int j = 0; j < col; j++) {
      cout<<c[i][j]<<b[i][j]<<"\t";
    }
    cout<<endl<<endl;
  }
  printLCS(x, y, row, col);
}


void LCS(char *x, char *y){
  int m, n;
```

```
m = strlen(x);

n = strlen(y);

seq = new char[n];


for(int i=0; i<m+1; i++)

{

  c[i][0] = '0';

  b[i][0] = ' ';

}

for(int j=0; j<n+1; j++)

{

  c[0][j] = '0';

  b[0][j] = ' ';

}


for(int i=1; i<m+1; i++)

  for( int j=1; j<n+1; j++)

  {

    if(x[i-1] == y[j-1]){

         c[i][j]=c[i-1][j-1] + 1;

         b[i][j] = '\\';

    }

    else if(c[i-1][j] >= c[i][j-1]){
            c[i][j]=c[i-1][j];

            b[i][j] = '|';

        }

        else{

            c[i][j]=c[i][j-1];

            b[i][j] = '-';

        }

      }

      printTable(x, y, m+1, n+1);


}


int main()

{


    int m, n;

    char y[10], x[10];

    cout<<"\nEnter the array \'X\' and \'Y\': \n\n";

    gets(x);

    gets(y);


    LCS(x, y);
```

```
  getch();

  return 0;

}
```

Output:



C:\Users\DELL\OneDrive\Desktop\codeblock\pro\bin\Debug\pro.exe

```
Enter the array 'X' and 'Y':

ABXYS
XYSABC

The graph for given Sequences:

        Y       X       Y       S       A       B       C

X       0       0       0       0       0       0       0

A       0       0|      0|      0|      1\      1-      1-

B       0       0|      0|      0|      1|      2\      2-

X       0       1\      1-      1-      1|      2|      2|

Y       0       1|      2\      2-      2-      2|      2|

S       0       1|      2|      3\      3-      3-      3-

------------------

LCS:    {   X   Y   S   }
Process returned 0 (0x0)   execution time : 24.906 s
Press any key to continue.
```

7. Implement Breadth-First Search in a graph

Solution:

```
#include<iostream>

#include<conio.h>

using namespace std;


int graph[20][2], rows=-1, index = 0;


class node
```

```
  {
     public:
         int info;
         node *next;

     node(int data, node *ptr = NULL)
         {
           info = data;
           next = ptr;
         }
};

class queue
{
         node *head, *tail;

  public:
  queue()
          {
             head = NULL;
             tail = NULL;
          }
```

```
int giveHead()
{
 return head->info;
}
     void enqueue(int val)
{
 if(head != NULL)
 {
  tail->next = new node(val);
  tail = tail->next;
 }
 else
  head = tail = new node(val);
}

     int dequeue()
{
 int temp;
 if(head != NULL)
 {
   temp = head->info;
        head = head->next;
   return temp;
```

```
    }
    else
                return -1;
  }
} Q;


void printGraph()
{

 cout<<"\nYour Graph: ";
 cout<<"\n\n  From     To\n\n";
  for(int i = 0; i < rows; i++) {
    for (int j = 0; j < 2; j++) {
      cout<<"    "<<graph[i][j]<<"\t";
    }
    cout<<endl;
  }
}

int checkPeresence(int a, int row, int col)
{
   for(int i=0; i<row; i++)
```

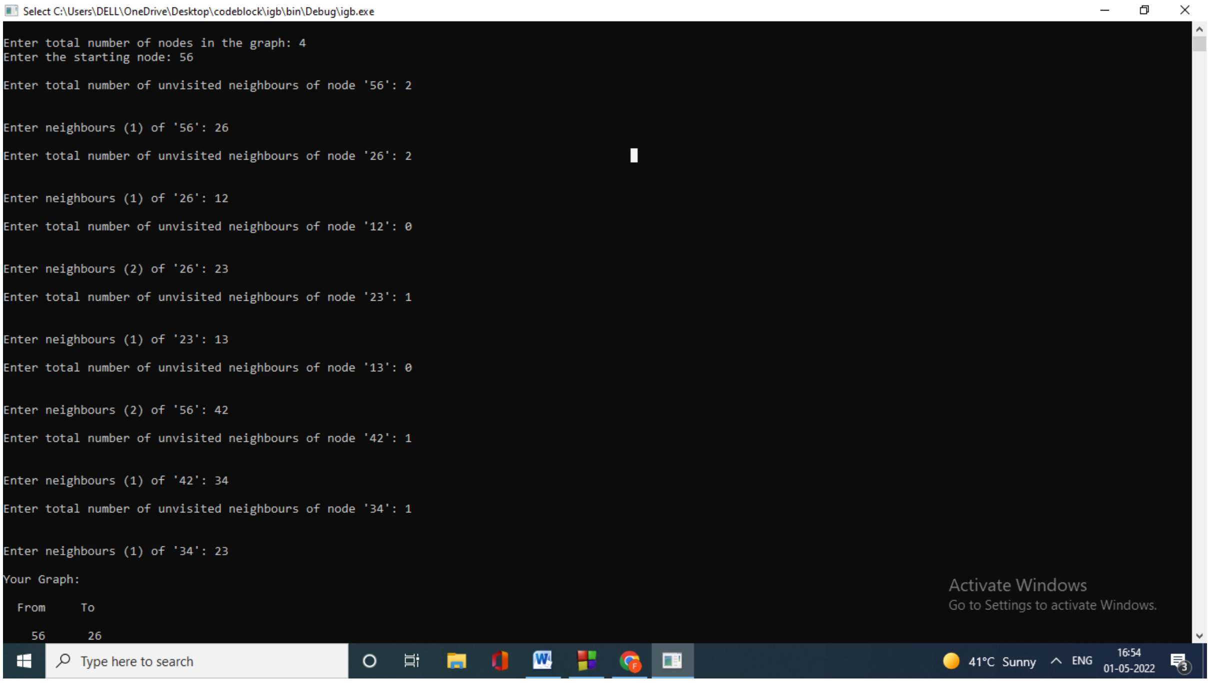
```
     if(graph[i][col] == a)
       return 1;
   return 0;
}

void drawGraph(int x, int i, int limit)
{
    int n, j;

    cout<<"\nEnter total number of unvisited neighbours of node \'"<<x<<"\': ";
    cin>>n;
    rows += n;

    if(n>0)
    {
      for (int j = i; j < i+n; j++) {
                graph[j][0] = x;
      }

      for (j = i; j < i+n; j++) {
           cout<<"\n\nEnter neighbours ("<<j-i+1<<") of \'"<<x<<"\': ";
           cin>>graph[j][1];
```

```
          if(checkPeresence(graph[j][1], rows, 0) == 0)
            drawGraph(graph[j][1], rows, limit);
      }
    }
}

void fill(int x)
{
  cout<<x<<"   ";
  Q.enqueue(x);
}

void BFS()
{
  while ((index < rows) && (Q.giveHead() == graph[index][0])) {
    if(checkPeresence(graph[index][1], index, 1) == 0)
      fill(graph[index][1]);
    index++;
  }
  if(index < rows)
  {
    do {
         if (Q.dequeue() == -1)
```

```
          return;
  } while(Q.giveHead() != graph[index][0]);

  BFS();
 }
 return;
}

int main()
{

  int t, start;
  rows = 0;

  cout<<"\nEnter total number of nodes in the graph: ";
  cin>>t;
  cout<<"Enter the starting node: ";
  cin>>start;

  drawGraph(start, 0, t-1);
  printGraph();

  getch();
```

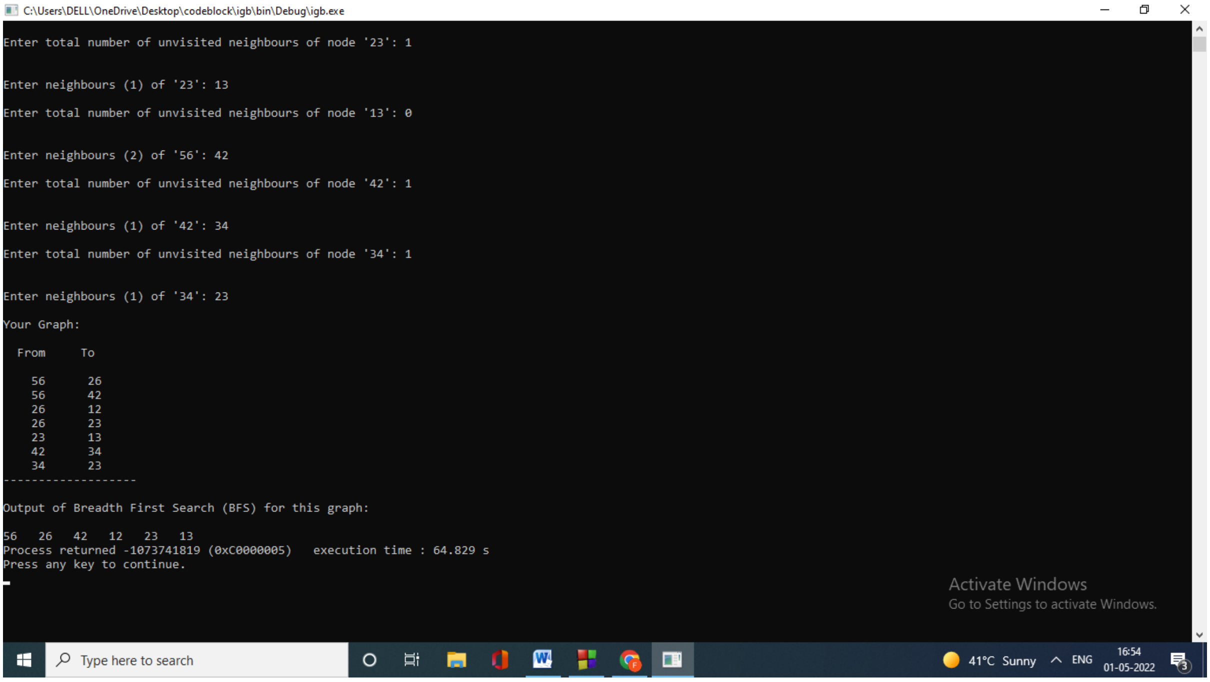
```
   cout<<"------------------\n\n"
              <<"Output of Breadth First Search (BFS) for this graph:"
              <<"\n\n";
   fill(start);
   BFS();


  getch();


  return 0;
}
```

Output:



Select C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
41°C Sunny
ENG
16:54
01-05-2022

```
Enter total number of nodes in the graph: 4
Enter the starting node: 56

Enter total number of unvisited neighbours of node '56': 2


Enter neighbours (1) of '56': 26

Enter total number of unvisited neighbours of node '26': 2


Enter neighbours (1) of '26': 12

Enter total number of unvisited neighbours of node '12': 0


Enter neighbours (2) of '26': 23

Enter total number of unvisited neighbours of node '23': 1


Enter neighbours (1) of '23': 13

Enter total number of unvisited neighbours of node '13': 0


Enter neighbours (2) of '56': 42

Enter total number of unvisited neighbours of node '42': 1


Enter neighbours (1) of '42': 34

Enter total number of unvisited neighbours of node '34': 1


Enter neighbours (1) of '34': 23

Your Graph:

  From     To

    56      26
```



```
C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Enter total number of unvisited neighbours of node '23': 1


Enter neighbours (1) of '23': 13

Enter total number of unvisited neighbours of node '13': 0


Enter neighbours (2) of '56': 42

Enter total number of unvisited neighbours of node '42': 1


Enter neighbours (1) of '42': 34

Enter total number of unvisited neighbours of node '34': 1


Enter neighbours (1) of '34': 23

Your Graph:

  From     To

    56      26
    56      42
    26      12
    26      23
    23      13
    42      34
    34      23
-------------------

Output of Breadth First Search (BFS) for this graph:

56   26   42   12   23   13
Process returned -1073741819 (0xC0000005)   execution time : 64.829 s
Press any key to continue.
```

8. Implement Depth-First Search in a graph

Solution:

```
#include<iostream>

#include<conio.h>



using namespace std;



int graph[20][3], rows=-1;

int vFlag[20][2], tNodes, index = 0;



class node

{

  public:
        int info;
        node *next;

  node(int data, node *ptr = NULL)
        {
          info = data;
          next = ptr;
        }
};

class stack
{
        node *top;

  public:
  stack()
        { top = NULL; }

  int givetop()
  {  return top->info;  }

  void push(int val)
  {
```

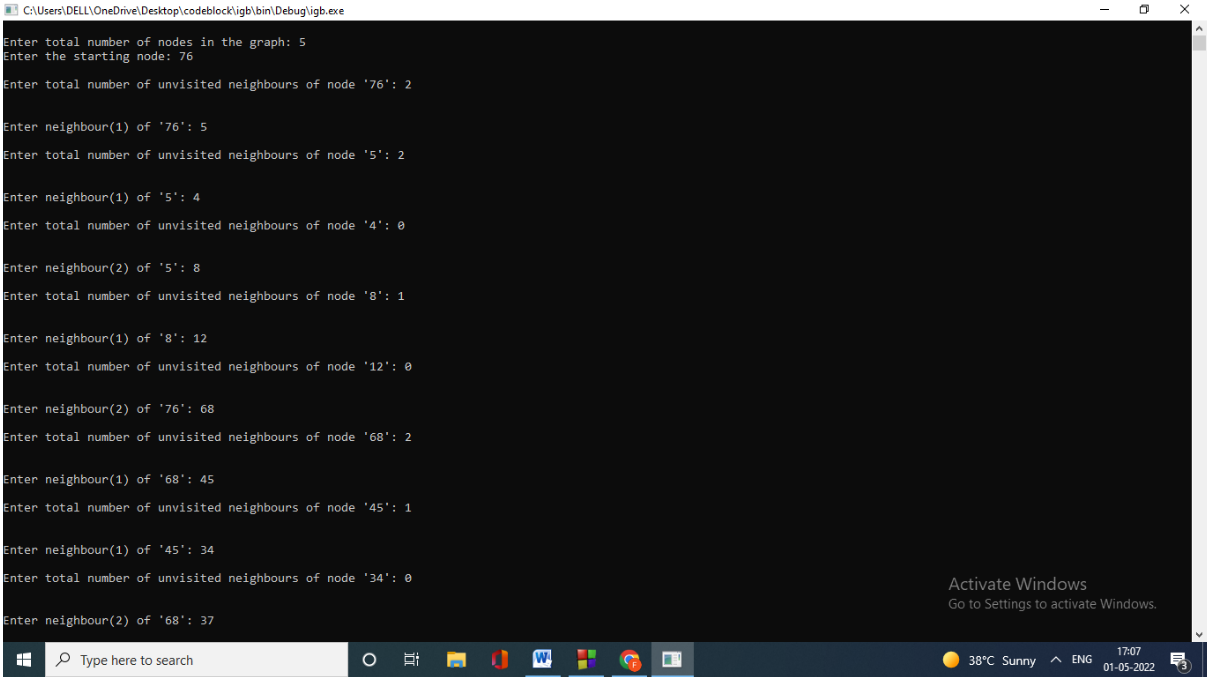
```
  if(top != NULL)
   top = new node(val, top);
  else
   top = new node(val);
 }

 void pop()
 {
  if(top != NULL)
        top = top->next;
  else
        return;
 }

}S;

int getFirstIndex(int n)
{
 for (int c = 0; c < rows; c++)
  if(graph[c][0] == n)
   return c;
 return -1;
}
```

```
void printGraph()
{

 cout<<"\nYour Graph: ";
 cout<<"\n\n  From     To\n\n";
  for(int i = 0; i < rows; i++) {
    for (int j = 0; j < 2; j++) {
      cout<<"    "<<graph[i][j]<<"\t";
    }
    cout<<endl;
  }
}

int checkPeresence(int a, int row, int col)
{
   for(int i=0; i<row; i++)
     if(graph[i][col] == a)
       return 1;
   return 0;
}

void drawGraph(int x, int i, int limit)
```

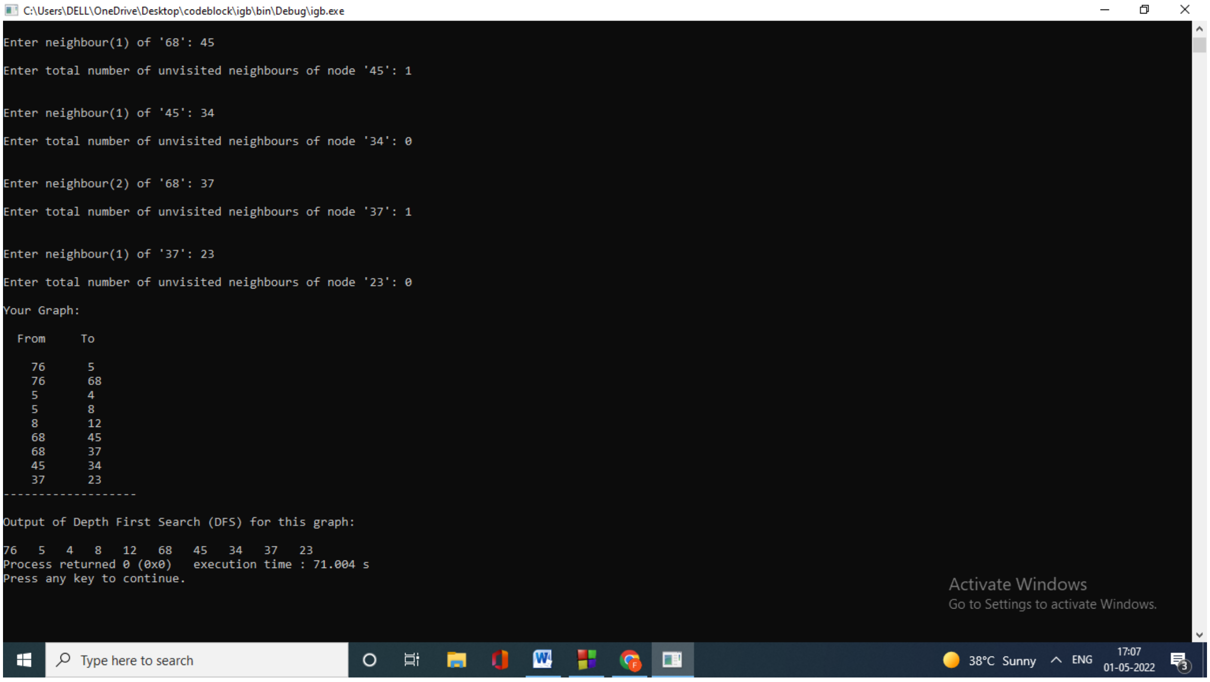
```
{
  int n, j,c;


  cout<<"\nEnter total number of unvisited neighbours of node \'"<<x<<"\': ";
  cin>>n;
  rows += n;


  if(n>0)
  {
   for (int j = i; j < i+n; j++) {
         graph[j][0] = x;
   }


   for (j = i; j < i+n; j++) {
        cout<<"\n\nEnter neighbour("<<j-i+1<<") of \'"<<x<<"\': ";
        cin>>graph[j][1];


     for(int c=0; (c<index) && (graph[j][1] != vFlag[c][0]); c++);
        if(c == index)
         vFlag[index++][0] = graph[j][1];


        if(checkPeresence(graph[j][1], rows, 0) == 0)
         drawGraph(graph[j][1], rows, limit);
```

```
      }
    }
}

void markVisited(int val) {
  for(int i=0; i<tNodes; i++)
    if(vFlag[i][0] == val)
    {
      vFlag[i][1] = 1;
      return;
    }
}

int isVisited(int x){
  for (int i = 0; i < tNodes; i++)
    if (vFlag[i][0] == x){
      return (vFlag[i][1] == 1) ? 0 : -1;
    }
  return -1;
}

void fill(int x)
{
  cout<<x<<"   ";
  markVisited(x);
  S.push(x);
}

void DFS(int v, int i)
{
  int j, temp;
  if(isVisited(v == -1))
    fill(v);

  while((S.givetop() == graph[i][0])&&(i<rows)){
    if(isVisited(graph[i][1]) == -1){
      j = getFirstIndex(graph[i][1]);
      if(j != -1){
            DFS(graph[i][1], j);
      }
      else{
            fill(graph[i][1]);
            S.pop();
      }
    }
    i++;
  }
  S.pop();
}

int main()
{

    int start, t=0;
    rows = 0;

    cout<<"\nEnter total number of nodes in the graph: ";
    cin>>tNodes;
    cout<<"Enter the starting node: ";
    cin>>start;
    vFlag[index++][0] = start;

    drawGraph(start, 0, tNodes-1);
    printGraph();

    for (int c = 0; c < tNodes; c++) {
                vFlag[c][1] = 0;
    }
  getch();


  cout<<"------------------\n\n"
         <<"Output of Depth First Search (DFS) for this graph:"
         <<"\n\n";


  DFS(start, 0);


 getch();


 return 0;
}
```

Output:


C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
38°C Sunny
ENG
17:07
01-05-2022

```
Enter total number of nodes in the graph: 5
Enter the starting node: 76

Enter total number of unvisited neighbours of node '76': 2


Enter neighbour(1) of '76': 5

Enter total number of unvisited neighbours of node '5': 2


Enter neighbour(1) of '5': 4

Enter total number of unvisited neighbours of node '4': 0


Enter neighbour(2) of '5': 8

Enter total number of unvisited neighbours of node '8': 1


Enter neighbour(1) of '8': 12

Enter total number of unvisited neighbours of node '12': 0


Enter neighbour(2) of '76': 68

Enter total number of unvisited neighbours of node '68': 2


Enter neighbour(1) of '68': 45

Enter total number of unvisited neighbours of node '45': 1


Enter neighbour(1) of '45': 34

Enter total number of unvisited neighbours of node '34': 0


Enter neighbour(2) of '68': 37
```



C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
38°C Sunny
ENG
17:07
01-05-2022

```
Enter neighbour(1) of '68': 45

Enter total number of unvisited neighbours of node '45': 1


Enter neighbour(1) of '45': 34

Enter total number of unvisited neighbours of node '34': 0


Enter neighbour(2) of '68': 37

Enter total number of unvisited neighbours of node '37': 1


Enter neighbour(1) of '37': 23

Enter total number of unvisited neighbours of node '23': 0

Your Graph:

  From     To

    76      5
    76      68
    5       4
    5       8
    8       12
    68      45
    68      37
    45      34
    37      23
-------------------

Output of Depth First Search (DFS) for this graph:

76   5   4   8   12   68   45   34   37   23
Process returned 0 (0x0)   execution time : 71.004 s
Press any key to continue.
```

9. Write a program to determine the minimum spanning tree of a graph

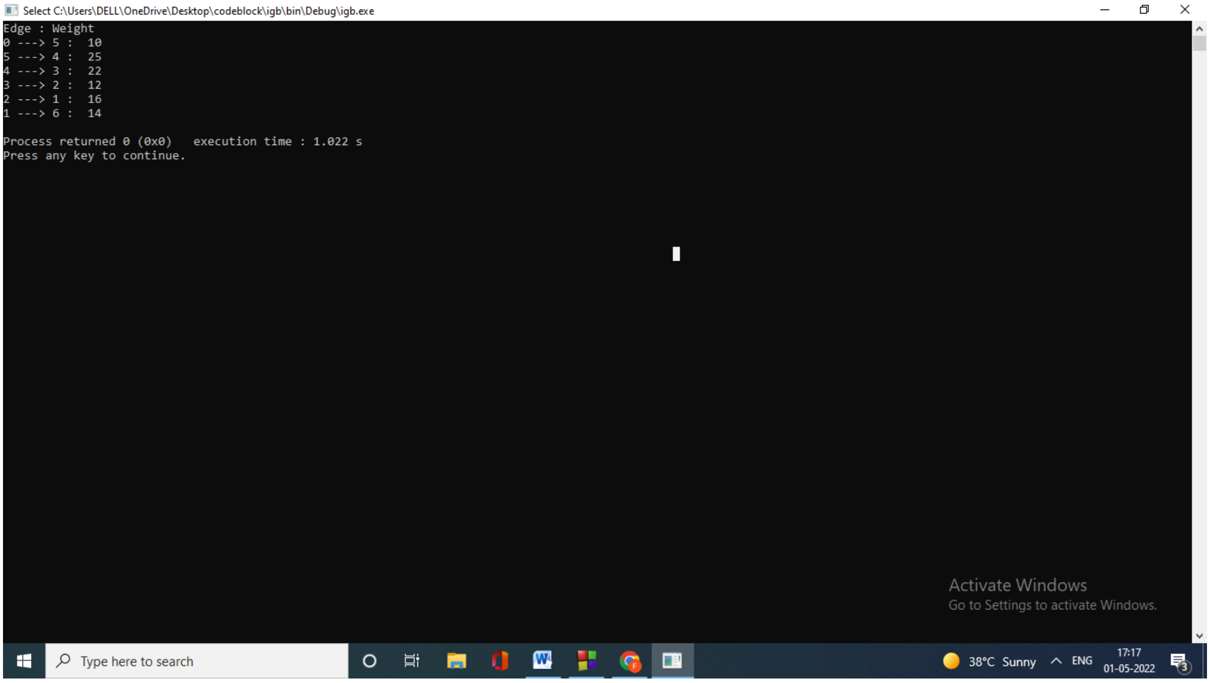
Solution:

```
 #include <iostream>

 #include<bits/stdc++.h>

 #include <cstring>

 using namespace std;

#define V 7


int main () {

int G[V][V] = {

         {0,28,0,0,0,10,0},

         {28,0,16,0,0,0,14},
           {0,16,0,12,0,0,0},

           {0,0,12,22,0,18},

           {0,0,0,22,0,25,24},

           {10,0,0,0,25,0,0},

           {0,14,0,18,24,0,0}

 };


  int edge;

  int visit[V];


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

   visit[i]=false;

}


  edge = 0;

  visit[0] = true;

  int x;

  int y;




  cout << "Edge" << " : " << "Weight";

  cout << endl;

  while (edge < V - 1) {
```

```
int min = INT_MAX;
x = 0;
y = 0;

for (int i = 0; i < V; i++) {
  if (visit[i]) {
     for (int j = 0; j < V; j++) {
      if (!visit[j] && G[i][j]) {
         if (min > G[i][j]) {
            min = G[i][j];
            x = i;
            y = j;
         }

      }
   }
  }
}
cout << x <<  " ---> " << y << " :  " << G[x][y];
cout << endl;
visit[y] = true;
edge++;
```

```
    }

  return 0;

}
```

Output:



Select C:\Users\DELL\OneDrive\Desktop\codeblock\igb\bin\Debug\igb.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
38°C Sunny
ENG
17:17
01-05-2022

```
Edge : Weight
0 ---> 5 :  10
5 ---> 4 :  25
4 ---> 3 :  22
3 ---> 2 :  12
2 ---> 1 :  16
1 ---> 6 :  14

Process returned 0 (0x0)   execution time : 1.022 s
Press any key to continue.
```