National Institute of Technology
durgapur

Design and Analysis of Algorithms

CSS-551

Lab Report

Prasun Kumar Bhuin

18CS8028

30 April, 2021

Contents

1 Exponential vs Polynomial Running time solution Fibonacci Sequence **2**
1.1 Code . . . 2
1.2 Output . . . 3

2 Heap and Priority Queue **4**
2.1 Code . . . 4
2.2 Output . . . 6

3 Counting Sort **7**
3.1 Code . . . 7
3.2 Output . . . 8

4 Merge Sort **9**
4.1 Code . . . 9
4.2 Output . . . 11

5 Fractional Knapsack Problem **12**
5.1 Code . . . 12
5.2 Output . . . 13

6 Longest Common Subsequence Algorithm **14**
6.1 Question . . . 14
6.2 Code . . . 14
6.3 Output . . . 15

7 Depth First Search in a Graph **16**
7.1 Question . . . 16
7.2 Code . . . 16
7.3 Output . . . 17

8 Finding Kth smallest element in worst case linear time **18**
8.1 Code . . . 18
8.2 Output . . . 19

9 Randomised Quick Sort **20**
9.1 Code . . . 20
9.2 Output . . . 21

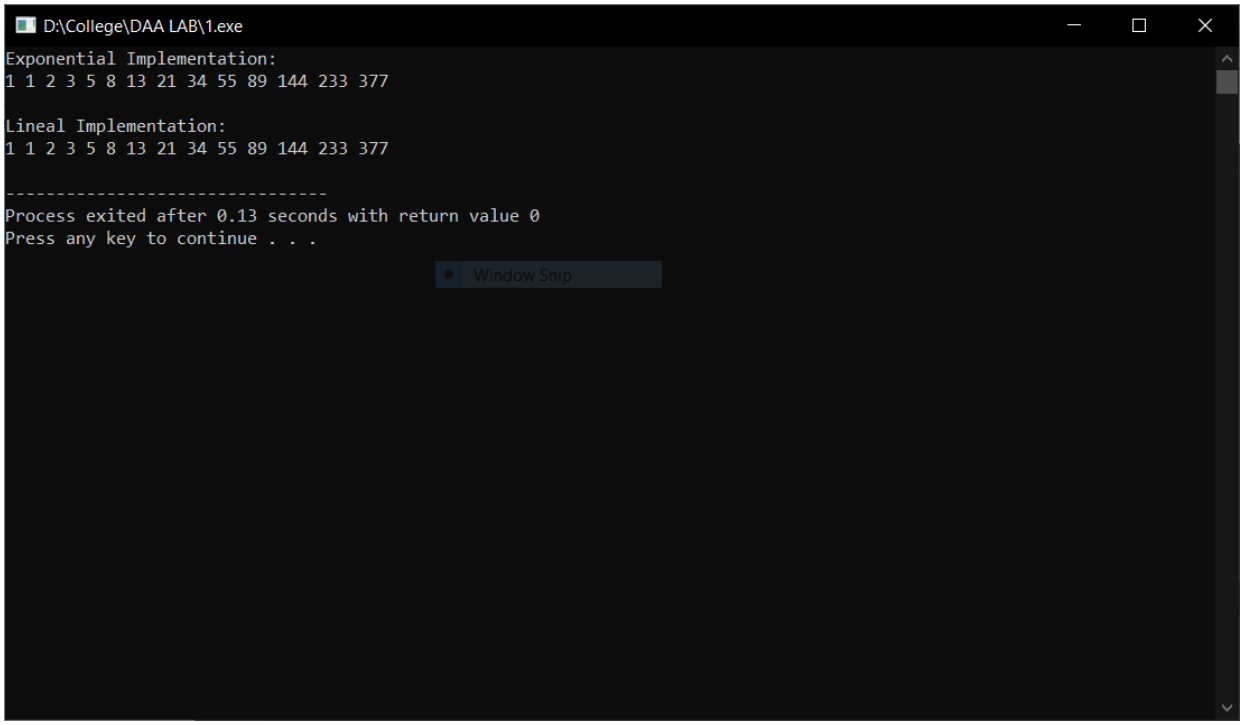
10 Convex Hull computation in 2D **22**
10.1 Code . . . 22
10.2 Output . . . 23

1 Exponential vs Polynomial Running time solution Fibonacci Sequence

1.1 Code

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

double fibonacciexpo(int n){ // Exponential algorithm
 if (n==1 || n==0){
  return n;
 }
 return fibonacciexpo(n-1) + fibonacciexpo(n-2);
}

double Fib[1000000];
double fibonaccilinear(int n){
 if (n==1 || n==0){
  return n;
 }
 if (Fib[n] != 0){
  return Fib[n];
 }
 Fib[n] = fibonaccilinear(n-1) + fibonaccilinear(n -2);
 return Fib[n];
}

int main(){

 int i;
 cout << "Exponential Implementation: \n";
 for (i=1; i<15; i++){
  cout << fibonacciexpo(i) << " ";
 }
 cout << "\n";

 cout << "\nLineal Implementation: \n";
 for (i=1; i<15; i++){
  cout << fibonaccilinear(i) << " ";
 }
 cout << "\n";


 return 0;
}
```

1.2 Output


D:\College\DAA LAB\1.exe
Exponential Implementation:
1 1 2 3 5 8 13 21 34 55 89 144 233 377
Lineal Implementation:
1 1 2 3 5 8 13 21 34 55 89 144 233 377

Process exited after 0.13 seconds with return value 0
Press any key to continue . . .

2 Heap and Priority Queue

2.1 Code

```
#include <iostream>

using namespace std;

void Max_heapify(int arr[], int n, int i)
{
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < n && arr[l] > arr[largest])
        largest = l;

    if (r < n && arr[r] > arr[largest])
        largest = r;

    if (largest != i) {
        swap(arr[i], arr[largest]);

        Max_heapify(arr, n, largest);
    }
}

void Heapify(int arr[], int n)
{
    int startIdx = (n / 2) - 1;

    for (int i = startIdx; i >= 0; i--) {
        Max_heapify(arr, n, i);
    }
}

void Delete(int arr[], int n)
{
 if(n<=0)
 cout<<"Empty Heap\n";
 else
  for(int i=0; i<n; i++)
  {
   arr[i]=arr[i+1];
  }
}


void Display(int arr[], int n)
{
    cout << "Array representation of Heap is:\n";

    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}


int main()
{
```

```
 int arr[20],a,n=1,i;
 cout<<"Enter first element of the Priority Queue :";
 cin>>arr[0];
 while(a!=4)
 {
  cout<<"\nSelect Command:\n1 -> Insert\n2 -> Delete\n3 -> Display\n4 -> Exit\n";
  cin>>a;
  if(a==1)
  {
   cout<<"Insert Element:";
   cin>>i;
   arr[n++] = i;
   Heapify(arr, n);
  }
  if(a==2)
  {
   if(n>0)
    cout<<"Deleted Element is:"<<arr[0]<<"\n";
   Delete(arr, n);
   n--;
   if(n<0)
    n=0;
   Heapify(arr, n);
  }
  if(a==3)
  {
   Display(arr, n);
  }
 }

}
```

2.2 Output

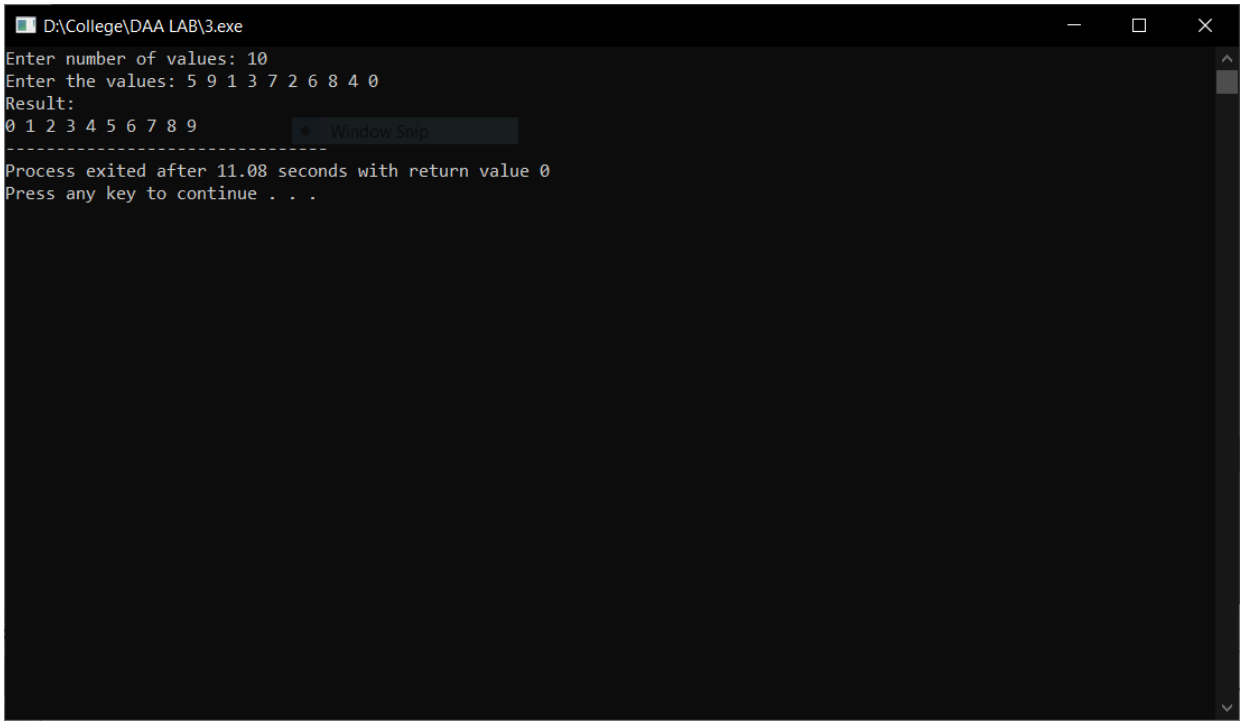
```
D:\College\DAA LAB\2.exe
Enter first element of the Priority Queue :5

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
1
Insert Element:3

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
1
Insert Element:7

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
3
Array representation of Heap is:
7 3 5

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
1
Insert Element:9

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
3
Array representation of Heap is:
9 7 5 3

Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
```

3 Counting Sort

3.1 Code

```
#include<bits/stdc++.h>
using namespace std;

void pusher(vector<int> &vec,int a, int n)
{
 for(int i=1;i<=n;i++)
 vec.push_back(a);
}

void countsort(int arr[],int n,int max)
{
 int count[max+1] = {0};
 int i;
 for(i=0;i<n;i++)
 count[arr[i]]++;

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

 vector<int> vec;
 pusher(vec,0,count[0]);

 for(i=1;i<=max;i++)
 {
  pusher(vec,i,count[i]-count[i-1]);
 }

 for(i=0;i<n;i++)
 arr[i] = vec[i];
}

int main()
{
 int n,max=0;
 cout<<"Enter number of values: ";
 cin>>n;

 int arr[n];

 cout<<"Enter the values: ";
 for(int i=0;i<n;i++)
 cin>>arr[i];

 for(int i=0;i<n;i++)
 if(arr[i]>max)
 max=arr[i];

 countsort(arr,n,max);

 cout<<"Result: "<<endl;
 for(int i=0;i<n;i++)
 cout<<arr[i]<<" ";

}
```

3.2 Output


D:\College\DAA LAB\3.exe
Enter number of values: 10
Enter the values: 5 9 1 3 7 2 6 8 4 0
Result:
0 1 2 3 4 5 6 7 8 9

Process exited after 11.08 seconds with return value 0
Press any key to continue . . .

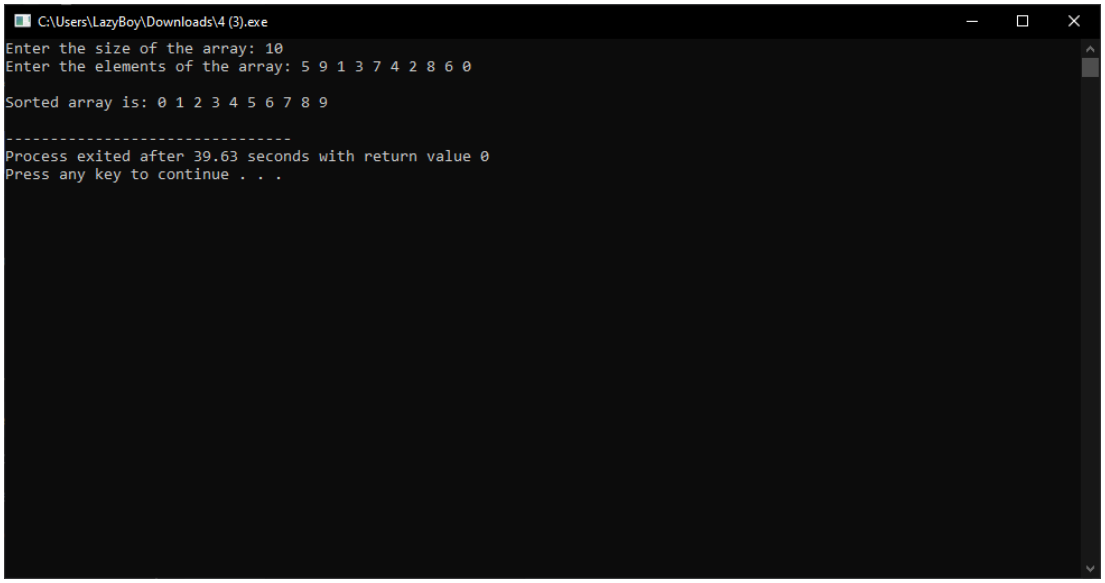
4 Merge Sort

4.1 Code

```

#include <iostream>
using namespace std;




void merge(int arr[], int l, int m, int r)
{
 int n1 = m - l + 1;
 int n2 = r - m;


 int L[n1], R[n2];


 for (int i = 0; i < n1; i++)
  L[i] = arr[l + i];
 for (int j = 0; j < n2; j++)
  R[j] = arr[m + 1 + j];




 int i = 0;


 int j = 0;


 int k = l;

 while (i < n1 && j < n2) {
  if (L[i] <= R[j]) {
   arr[k] = L[i];
   i++;
  }
  else {
   arr[k] = R[j];
   j++;
  }
  k++;
 }



 while (i < n1) {
  arr[k] = L[i];
  i++;
  k++;
 }



 while (j < n2) {
  arr[k] = R[j];
```

```
  j++;
  k++;
 }
}




void mergeSort(int arr[],int l,int r){
 if(l>=r){
  return;
 }
 int m =l+ (r-l)/2;
 mergeSort(arr,l,m);
 mergeSort(arr,m+1,r);
 merge(arr,l,m,r);
}



void printArray(int A[], int size)
{
 for (int i = 0; i < size; i++)
  cout << A[i] << " ";
    cout<<"\n";
}


int main()
{
 int n;
    cout<<"Enter the size of the array: ";
    cin>>n;
    int arr[n];
    cout<<"Enter the elements of the array: ";
    for(int i=0;i<n;i++)
        cin>>arr[i];


 mergeSort(arr, 0, n-1);

 cout << "\nSorted array is: ";
 printArray(arr, n);
 return 0;
}
```

4.2 Output


C:\Users\LazyBoy\Downloads\4 (3).exe
Enter the size of the array: 10
Enter the elements of the array: 5 9 1 3 7 4 2 8 6 0
Sorted array is: 0 1 2 3 4 5 6 7 8 9

Process exited after 39.63 seconds with return value 0
Press any key to continue . . .

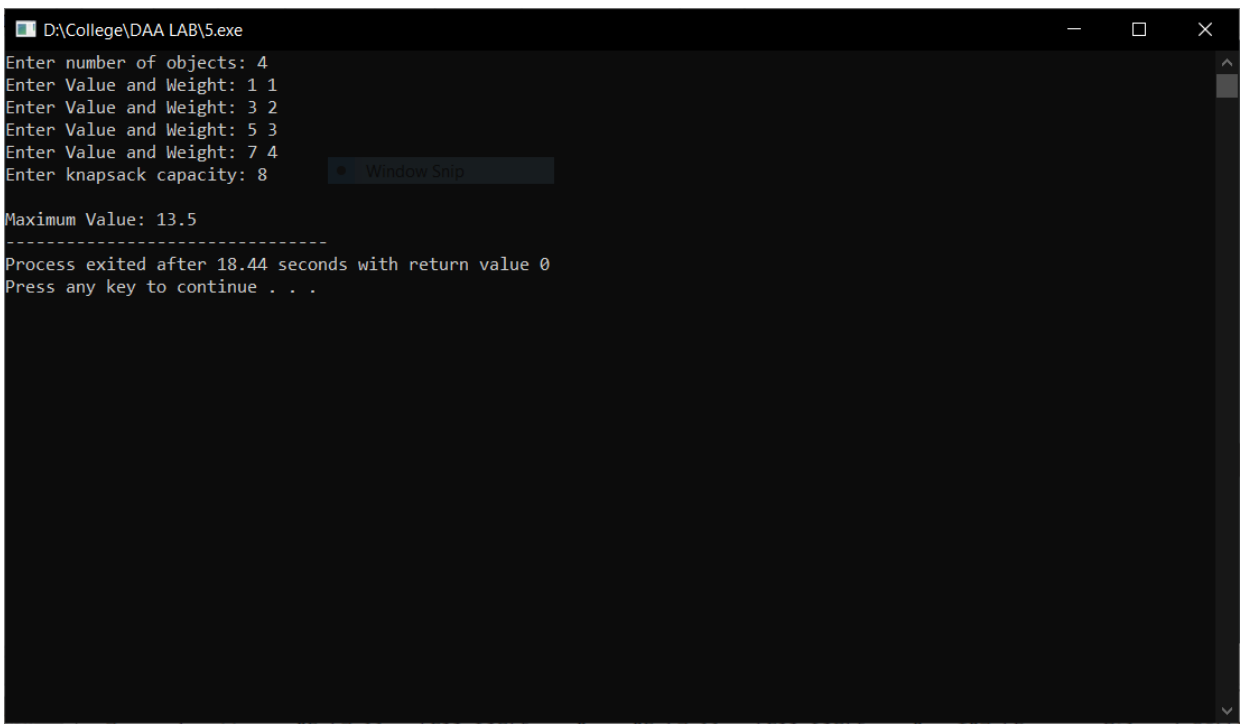
5 Fractional Knapsack Problem

5.1 Code

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

bool comparet( vector<int> a,vector<int> b){
 float ae = a[0] *1.0/ a[1];
 float be = b[0] *1.0/ b[1];
 return ae > be;
}

float fractionknapsack(int Values[], int Weights[], int kcapacity, int n){
 int i;

 vector<vector<int> > E;

 for (i=0; i<n; i++){
  vector<int> vec;
  vec.push_back(Values[i]);
  vec.push_back(Weights[i]);
//  cout << Values[i]<<Weights[i]<< " "<< vec[0]<<vec[1]<<"\n";
  E.push_back(vec);
 }



 sort(E.begin(), E.end(), comparet);



 float vsum = 0;
 float wrem = kcapacity, w, v;
 float p;
 i = 0;
 while (wrem > 0){
  if (i >= n) break; //all objects added to knapsack

  v = E[i][0];
  w = E[i][1];

  p = wrem/w;
  if (p>1) p=1.0;

  wrem -= w;
  vsum  += p*v;

  i+=1;
//  cout << vsum << " ";
 }
 return vsum;
}

int main(){
 int n;
 cout << "Enter number of objects: ";
 cin >> n;
```

```
 int Values[n], Weights[n], i=0;

 for (i=0; i<n; i++){
  cout <<"Enter Value and Weight: ";
  cin >> Values[i] >> Weights[i];
 }



 cout << "Enter knapsack capacity: ";
 int knapsack;
 cin >> knapsack;

// for (i=0; i<12; i++){
 float k =  fractionknapsack(Values, Weights, knapsack, n) ;
 cout << "\nMaximum Value: " << k;
//}

 return 0;
}
```

5.2 Output



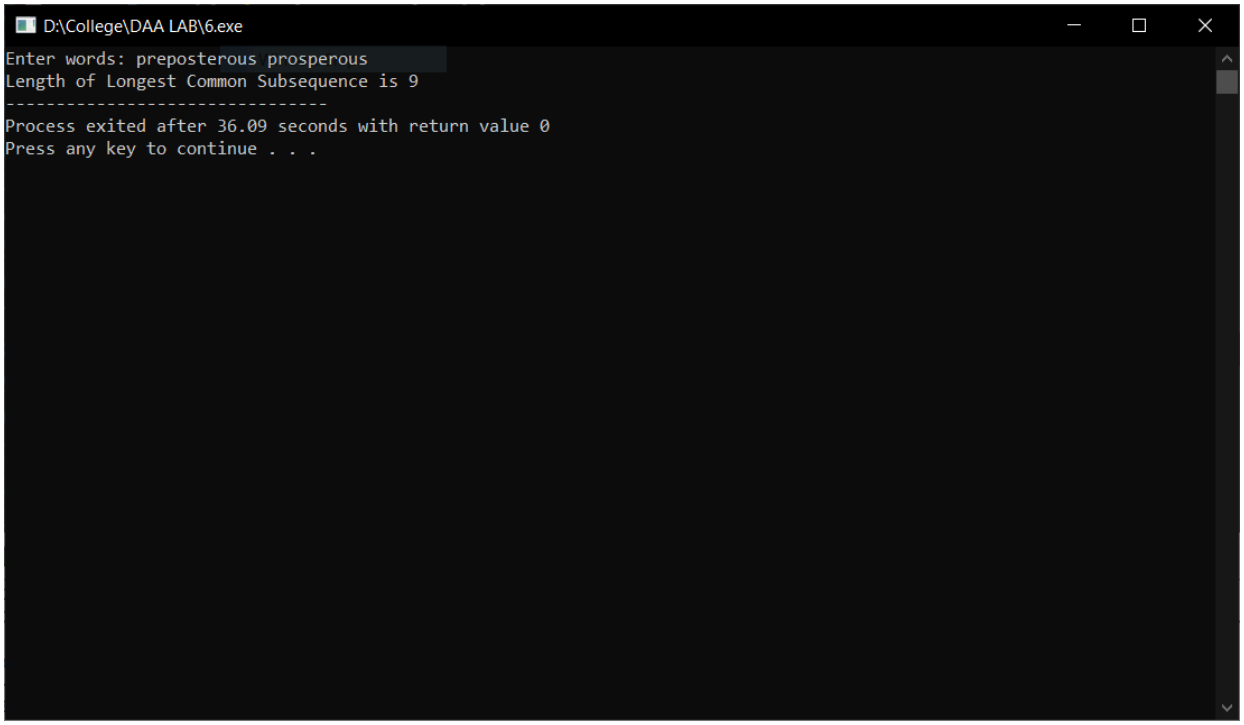
6 Longest Common Subsequence Algorithm

6.1 Question

6.2 Code

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

int longestcommonsubseq(string a, string b){
 int an = a.length(), bn = b.length();
 int M[an+1][bn+1] = {0};


 int i, j;


 for (i=0; i<an+1; i++){
  for (j=0; j<bn+1; j++){
   M[i][j] = 0;
  }
 }
 for (i=1; i<=an; i++){
  for (j=1; j<=bn; j++){
   M[i][j] = max(M[i][j-1], M[i-1][j]) + (a[i]==b[j]);
//    cout << M[i][j]<<"\t";
  }
//  cout <<"\n";
 }

 return M[an][bn];

}

int main(){
 string a, b;

 cout << "Enter words: ";

 cin >> a >> b;

 cout << "Length of Longest Common Subsequence is " << longestcommonsubseq(a, b);


 return 0;
}
```

6.3 Output


D:\College\DAA LAB\6.exe
Enter words: preposterous prosperous
Length of Longest Common Subsequence is 9

Process exited after 36.09 seconds with return value 0
Press any key to continue . . .

7 Depth First Search in a Graph

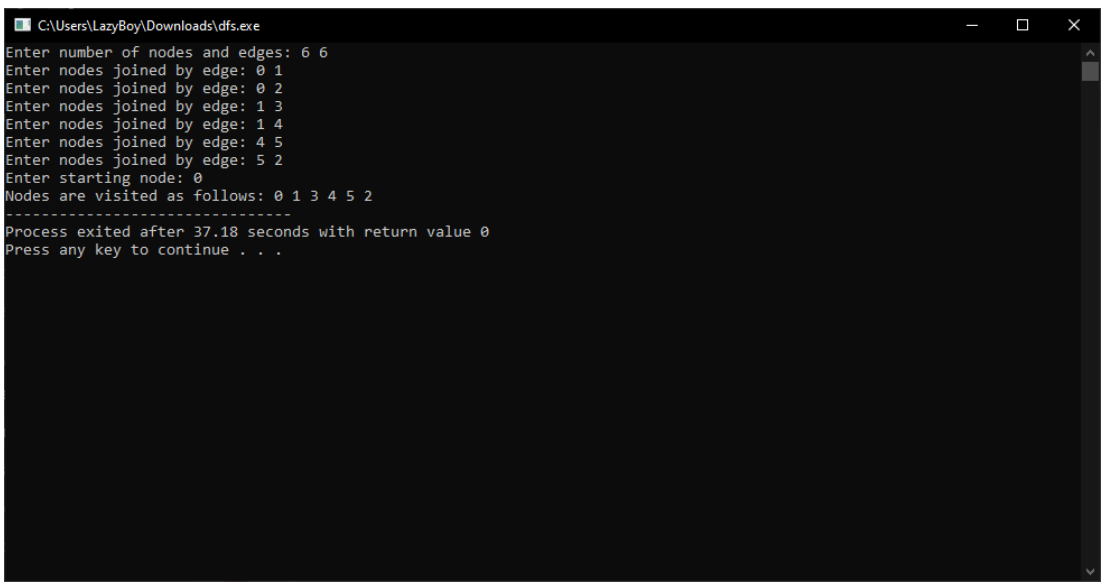
7.1 Question

7.2 Code

```
#include <bits/stdc++.h>
using namespace std;

class Graph {
 int v;
 int e;
 int** adj;

public:
 Graph(int v, int e);
 void addEdge(int start, int e);
 void DFS(int start, vector<bool>& visited);
};

Graph::Graph(int v, int e)
{
 this->v = v;
 this->e = e;
 adj = new int*[v];
 for (int row = 0; row < v; row++) {
  adj[row] = new int[v];
  for (int column = 0; column < v; column++) {
   adj[row][column] = 0;
  }
 }
}

void Graph::addEdge(int start, int e)
{
 adj[start][e] = 1;
 adj[e][start] = 1;
}

void Graph::DFS(int start, vector<bool>& visited)
{
 cout << start << " ";
 visited[start] = true;
 for (int i = 0; i < v; i++) {
  if (adj[start][i] == 1 && (!visited[i])) {
   DFS(i, visited);
  }
 }
}

int main()
{
 int v, e;
    cout << "Enter number of nodes and edges: ";
    cin>>v>>e;

 Graph G(v, e);

    int a,b;
    for(int i=0;i<e;i++){
```

```
        cout << "Enter nodes joined by edge: ";
        cin>>a>>b;
        G.addEdge(a,b);
    }
 vector<bool> visited(v, false);

    int s;
    cout << "Enter starting node: ";
    cin>>s;

    cout << "Nodes are visited as follows: ";

 G.DFS(s, visited);
}
```

7.3 Output



8 Finding Kth smallest element in worst case linear time

8.1 Code

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,k;
    cout<<"Enter limit: ";
    cin>>n;
    int arr[n];
    cout<<"Enter values: ";
    for(int i=0;i<n;i++)
    cin>>arr[i];

    cout<<"Enter value of k: ";
    cin>>k;

    int max = 0;
    for(int i=0;i<n;i++)
    if(arr[i]>max)
    max=arr[i];

    int count[max+1]={0};
    for(int i=0;i<n;i++)
    count[arr[i]]++;

    int sum=0,res;
    for(int i=0;i<=max;i++)
    {
        sum+=count[i];
        if(sum>=k)
        {
            res=i;
            break;
        }
    }
    cout<<"kth smallest element is: "<<res<<endl;
    return 0;
}
```

8.2 Output

```
D:\College\DAA LAB\8.exe
Enter limit: 10
Enter values: 55 99 11 33 77 44 66 22 88 00
Enter value of k: 7
kth smallest element is: 66

--------------------------------
Process exited after 36.51 seconds with return value 0
Press any key to continue . . .
```

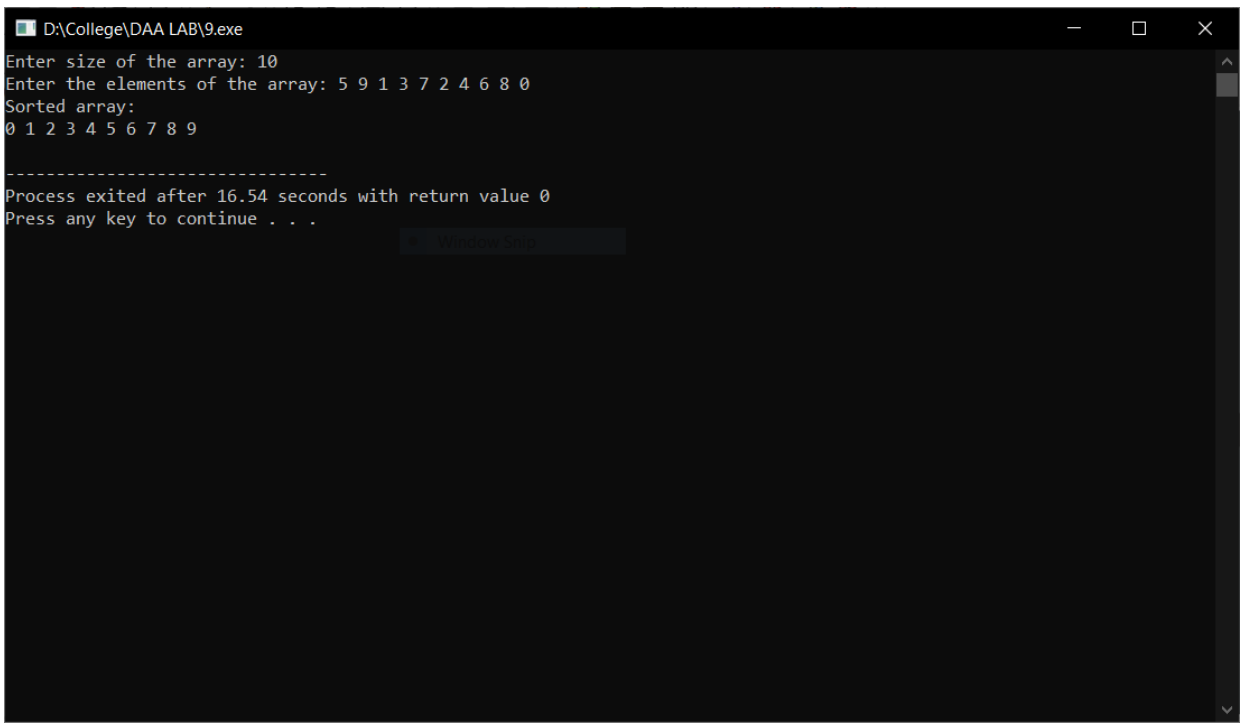
9 Randomised Quick Sort

9.1 Code

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


int partition(int arr[], int low, int high)
{
    int pivot = arr[high];

    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)
    {

        if (arr[j] <= pivot) {

            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return (i + 1);
}


int partition_r(int arr[], int low, int high)
{
    int random = low + rand() % (high - low);

    swap(arr[random], arr[high]);

    return partition(arr, low, high);
}


void quickSort(int arr[], int low, int high)
{
    if (low < high) {

        int pi = partition_r(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}


void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main()
```

```
{
    int arr[20],i,n;
    cout<<"Enter size of the array: ";
    cin>>n;
    cout<<"Enter the elements of the array: ";
    for(i=0; i<n; i++)
     cin>>arr[i];
    quickSort(arr, 0, n - 1);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

9.2 Output



10 Convex Hull computation in 2D

10.1 Code

```
#include <bits/stdc++.h>
using namespace std;

struct Point{
    int x, y;
};

int orientation(Point p, Point q, Point r){
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;
    return (val > 0)? 1: 2;
}


void convexHull(Point points[], int n){
    if (n < 3) return;

    vector<Point> hull;

    int l = 0;
    for (int i = 1; i < n; i++)
        if (points[i].x < points[l].x)
            l = i;

    int p = l, q;
    do
    {
        hull.push_back(points[p]);
        q = (p+1)%n;
        for (int i = 0; i < n; i++){
           if (orientation(points[p], points[i], points[q]) == 2)
               q = i;
        }

        p = q;

    } while (p != l);

    for (int i = 0; i < hull.size(); i++)
        cout << "(" << hull[i].x << ", "<< hull[i].y << ")\n";
}

int main(){
 int num_points;
    cout << "Enter the number of Points: ";
    cin >> num_points;

    if (num_points < 3)
    {
        cout << "\nConvex Hull is not possible\n";
        return 0;
    }


```

```
    Point XY_plane[num_points];

    for (int i = 0; i < num_points; i++)
    {
        printf("Point P%d: ", i + 1);
        cin >> XY_plane[i].x >> XY_plane[i].y;
    }

 cout << "Convex Hull passes through the following: \n";
    convexHull(XY_plane, num_points);
    return 0;
}
```

10.2 Output