

NATIONAL INSTITUTE OF TECHNOLOGY
DURGAPUR

DESIGN AND ANALYSIS OF ALGORITHMS

CSS-551

Lab Report

PRASUN KUMAR BHUIN

18CS8028

30 April, 2021

Contents

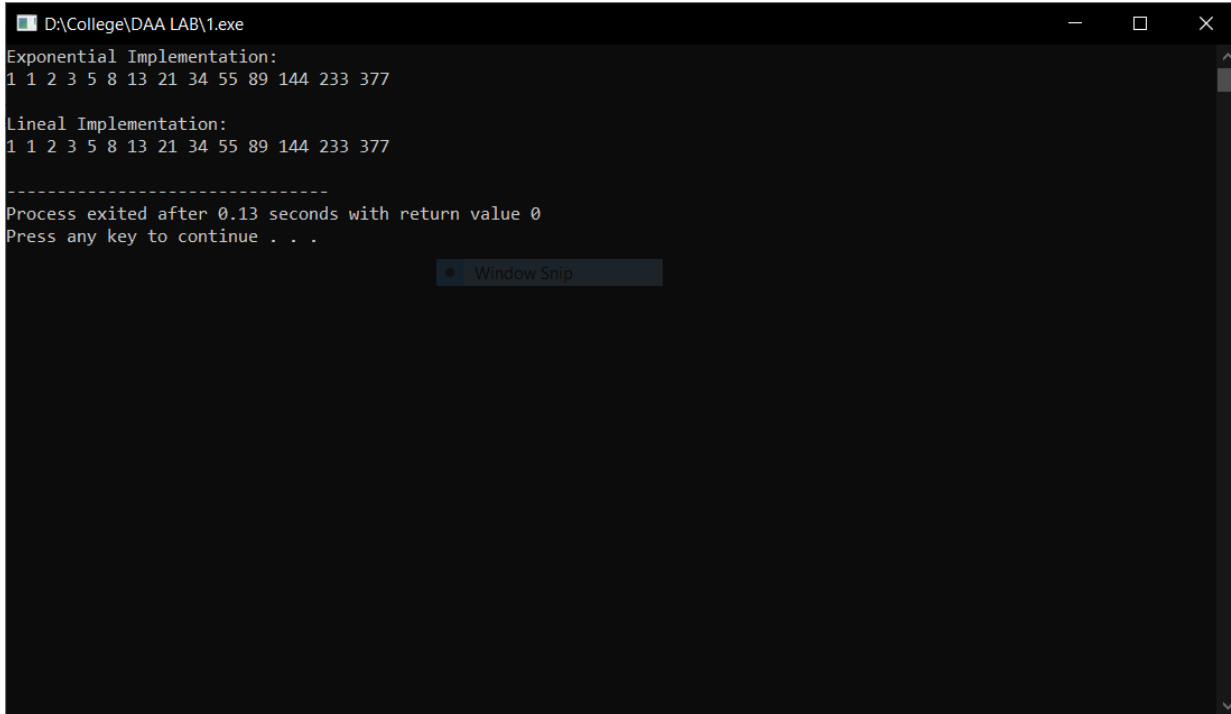
1	Exponential vs Polynomial Running time solution Fibonacci Sequence	2
1.1	Code	2
1.2	Output	3
2	Heap and Priority Queue	4
2.1	Code	4
2.2	Output	6
3	Counting Sort	7
3.1	Code	7
3.2	Output	8
4	Merge Sort	9
4.1	Code	9
4.2	Output	11
5	Fractional Knapsack Problem	12
5.1	Code	12
5.2	Output	13
6	Longest Common 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

```
1 #include <iostream>
2 using namespace std;
3
4 double fibonacciexpo(int n){ // Exponential algorithm
5     if (n==1 || n==0){
6         return n;
7     }
8     return fibonacciexpo(n-1) + fibonacciexpo(n-2);
9 }
10
11 double Fib[1000000];
12 double fibonaccilinear(int n){
13     if (n==1 || n==0){
14         return n;
15     }
16     if (Fib[n] != 0){
17         return Fib[n];
18     }
19     Fib[n] = fibonaccilinear(n-1) + fibonaccilinear(n-2);
20     return Fib[n];
21 }
22
23 int main(){
24
25     int i;
26     cout << "Exponential Implementation: \n";
27     for (i=1; i<15; i++){
28         cout << fibonacciexpo(i) << " ";
29     }
30     cout << "\n";
31
32     cout << "\nLineal Implementation: \n";
33     for (i=1; i<15; i++){
34         cout << fibonaccilinear(i) << " ";
35     }
36     cout << "\n";
37
38
39     return 0;
40 }
```

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

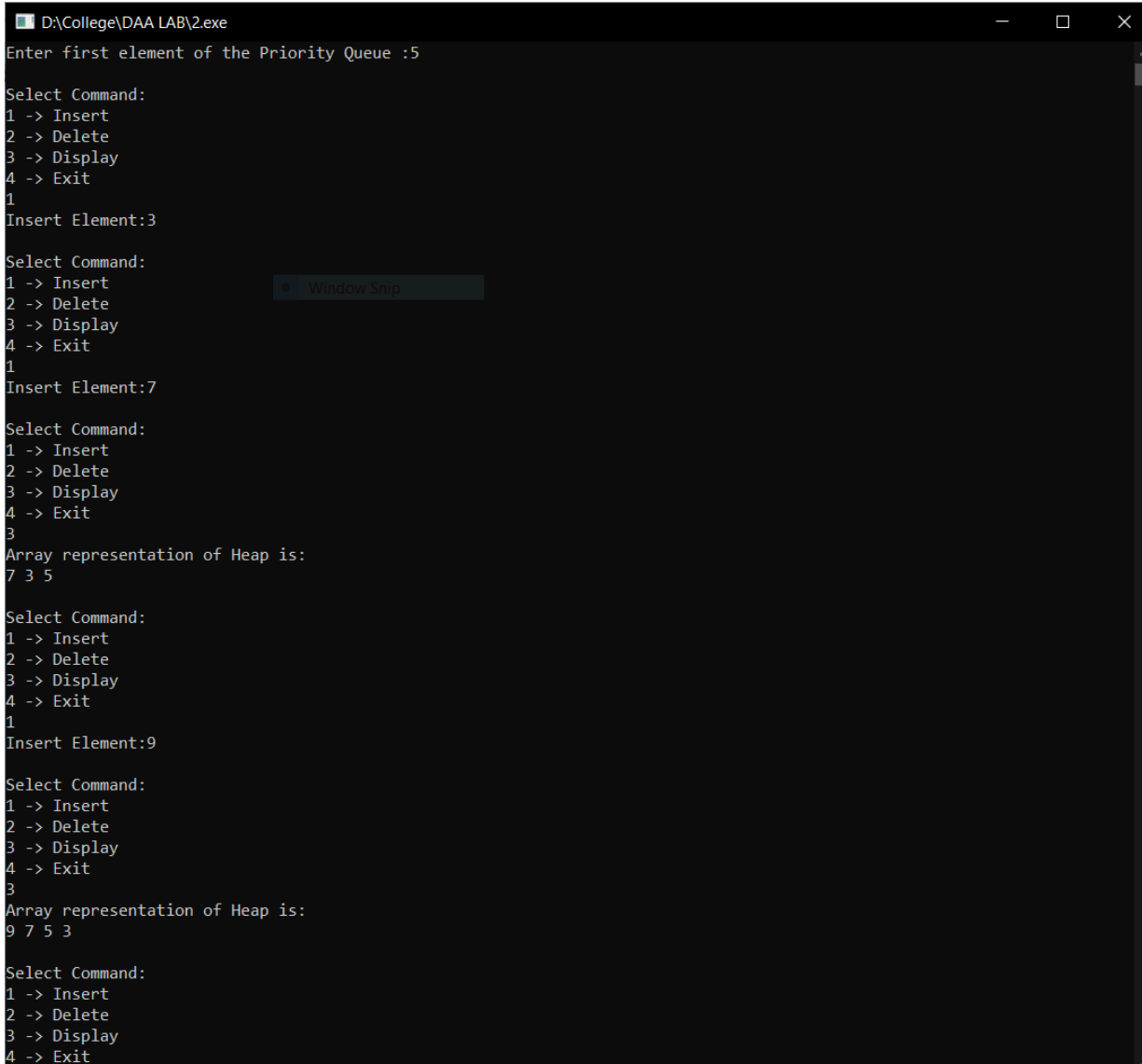
```
1 #include <iostream>
2
3 using namespace std;
4
5 void Max_heapify(int arr[], int n, int i)
6 {
7     int largest = i;
8     int l = 2 * i + 1;
9     int r = 2 * i + 2;
10
11     if (l < n && arr[l] > arr[largest])
12         largest = l;
13
14     if (r < n && arr[r] > arr[largest])
15         largest = r;
16
17     if (largest != i) {
18         swap(arr[i], arr[largest]);
19
20         Max_heapify(arr, n, largest);
21     }
22 }
23
24 void Heapify(int arr[], int n)
25 {
26     int startIdx = (n / 2) - 1;
27
28     for (int i = startIdx; i >= 0; i--) {
29         Max_heapify(arr, n, i);
30     }
31 }
32
33 void Delete(int arr[], int n)
34 {
35     if (n <= 0)
36         cout << "Empty Heap\n";
37     else
38         for (int i = 0; i < n; i++)
39             {
40                 arr[i] = arr[i + 1];
41             }
42 }
43
44
45 void Display(int arr[], int n)
46 {
47     cout << "Array representation of Heap is:\n";
48
49     for (int i = 0; i < n; ++i)
50         cout << arr[i] << " ";
51     cout << "\n";
52 }
53
54
55 int main()
56 {
```

```

57  int arr[20],a,n=1,i;
58  cout<<"Enter first element of the Priority Queue :";
59  cin>>arr[0];
60  while(a!=4)
61  {
62      cout<<"\nSelect Command:\n1 -> Insert\n2 -> Delete\n3 -> Display\n4 -> Exit\n";
63      cin>>a;
64      if(a==1)
65      {
66          cout<<"Insert Element:";
67          cin>>i;
68          arr[n++] = i;
69          Heapify(arr , n);
70      }
71      if(a==2)
72      {
73          if(n>0)
74              cout<<"Deleted Element is:"<<arr[0]<<"\n";
75          Delete(arr , n);
76          n--;
77          if(n<0)
78              n=0;
79          Heapify(arr , n);
80      }
81      if(a==3)
82      {
83          Display(arr , n);
84      }
85  }
86
87 }

```

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

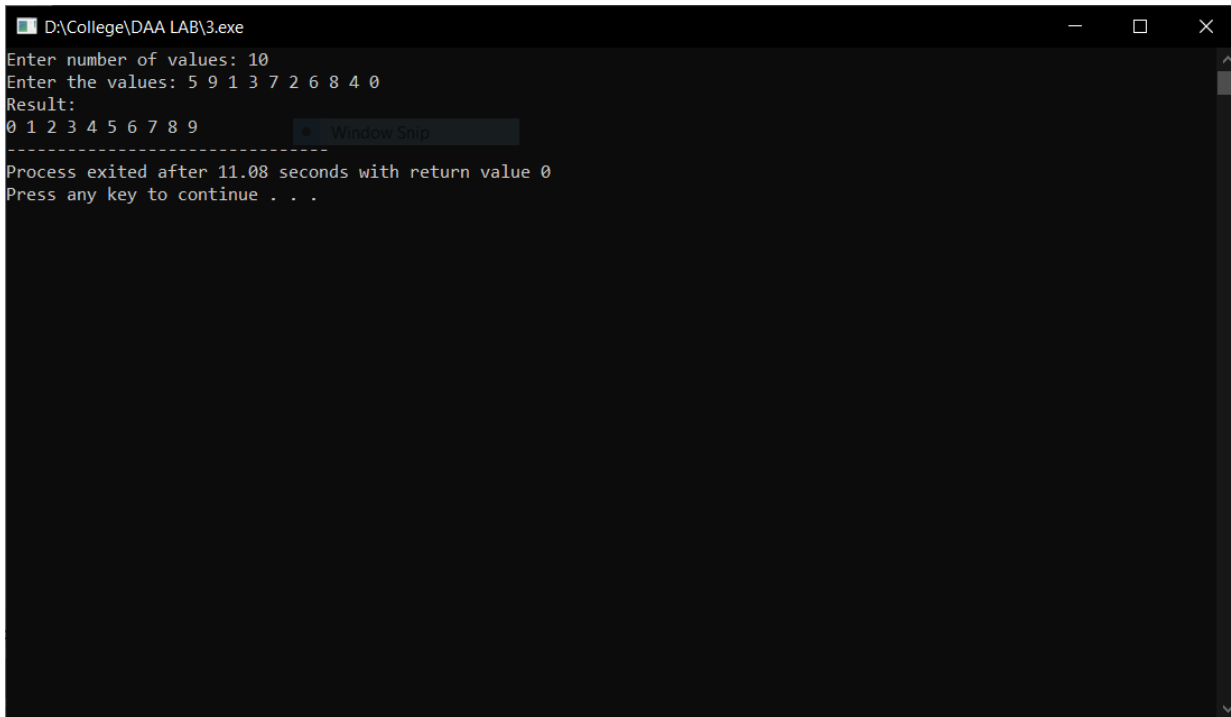
Select Command:
1 -> Insert
2 -> Delete
3 -> Display
4 -> Exit
```

3 Counting Sort

3.1 Code

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 void pusher(vector<int> &vec, int a, int n)
5 {
6     for(int i=1; i<=n; i++)
7         vec.push_back(a);
8 }
9
10 void countsort(int arr[], int n, int max)
11 {
12     int count[max+1] = {0};
13     int i;
14     for(i=0; i<n; i++)
15         count[arr[i]]++;
16
17     for(i=1; i<=max; i++)
18         count[i] += count[i-1];
19
20     vector<int> vec;
21     pusher(vec, 0, count[0]);
22
23     for(i=1; i<=max; i++)
24     {
25         pusher(vec, i, count[i] - count[i-1]);
26     }
27
28     for(i=0; i<n; i++)
29         arr[i] = vec[i];
30 }
31
32 int main()
33 {
34     int n, max=0;
35     cout<<"Enter number of values: ";
36     cin>>n;
37
38     int arr[n];
39
40     cout<<"Enter the values: ";
41     for(int i=0; i<n; i++)
42         cin>>arr[i];
43
44     for(int i=0; i<n; i++)
45         if(arr[i]>max)
46             max=arr[i];
47
48     countsort(arr, n, max);
49
50     cout<<"Result: "<<endl;
51     for(int i=0; i<n; i++)
52         cout<<arr[i]<<" ";
53
54 }
```


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

```
1
2 #include <iostream>
3 using namespace std;
4
5
6
7
8 void merge(int arr[], int l, int m, int r)
9 {
10     int n1 = m - l + 1;
11     int n2 = r - m;
12
13
14     int L[n1], R[n2];
15
16
17     for (int i = 0; i < n1; i++)
18         L[i] = arr[l + i];
19     for (int j = 0; j < n2; j++)
20         R[j] = arr[m + 1 + j];
21
22
23
24
25     int i = 0;
26
27
28     int j = 0;
29
30
31     int k = l;
32
33     while (i < n1 && j < n2) {
34         if (L[i] <= R[j]) {
35             arr[k] = L[i];
36             i++;
37         }
38         else {
39             arr[k] = R[j];
40             j++;
41         }
42         k++;
43     }
44
45
46
47     while (i < n1) {
48         arr[k] = L[i];
49         i++;
50         k++;
51     }
52
53
54
55     while (j < n2) {
56         arr[k] = R[j];
```

```

57     j++;
58     k++;
59 }
60 }
61
62
63
64
65 void mergeSort(int arr[], int l, int r){
66     if(l>=r){
67         return;
68     }
69     int m = l + (r-l)/2;
70     mergeSort(arr, l, m);
71     mergeSort(arr, m+1, r);
72     merge(arr, l, m, r);
73 }
74
75
76
77 void printArray(int A[], int size)
78 {
79     for (int i = 0; i < size; i++)
80         cout << A[i] << " ";
81     cout<<"\n";
82 }
83
84
85 int main()
86 {
87     int n;
88     cout<<"Enter the size of the array: ";
89     cin>>n;
90     int arr[n];
91     cout<<"Enter the elements of the array: ";
92     for(int i=0;i<n;i++)
93         cin>>arr[i];
94
95
96     mergeSort(arr, 0, n-1);
97
98     cout << "\nSorted array is: ";
99     printArray(arr, n);
100     return 0;
101 }

```

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

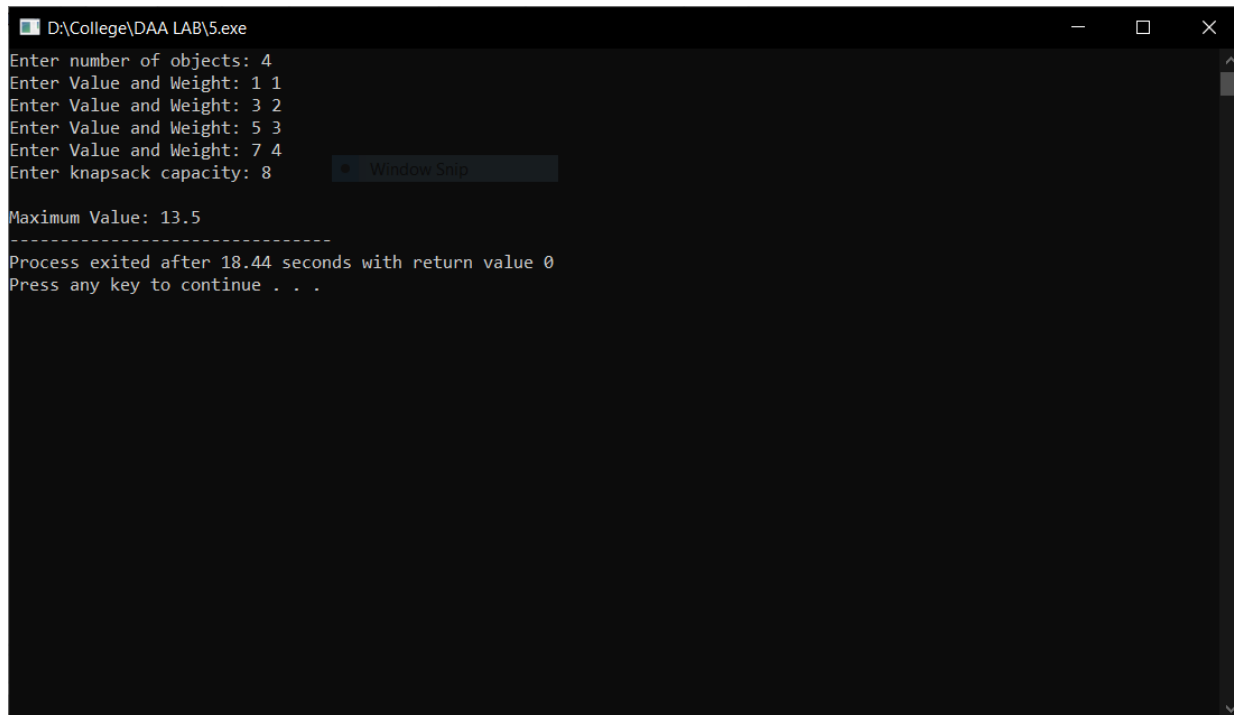
```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 using namespace std;
5
6 bool comparet( vector<int> a, vector<int> b){
7     float ae = a[0] *1.0/ a[1];
8     float be = b[0] *1.0/ b[1];
9     return ae > be;
10 }
11
12 float fractionknapsack(int Values[], int Weights[], int kcapacity, int n){
13     int i;
14
15     vector<vector<int>> E;
16
17     for (i=0; i<n; i++){
18         vector<int> vec;
19         vec.push_back(Values[i]);
20         vec.push_back(Weights[i]);
21         // cout << Values[i]<<Weights[i]<< " "<< vec[0]<<vec[1]<<"\n";
22         E.push_back(vec);
23     }
24
25
26
27     sort(E.begin(), E.end(), comparet);
28
29
30
31     float vsum = 0;
32     float wrem = kcapacity, w, v;
33     float p;
34     i = 0;
35     while (wrem > 0){
36         if (i >= n) break; //all objects added to knapsack
37
38         v = E[i][0];
39         w = E[i][1];
40
41         p = wrem/w;
42         if (p>1) p=1.0;
43
44         wrem -= w;
45         vsum += p*v;
46
47         i+=1;
48         // cout << vsum << " ";
49     }
50     return vsum;
51 }
52
53 int main(){
54     int n;
55     cout << "Enter number of objects: ";
56     cin >> n;
```

```

57
58 int Values[n], Weights[n], i=0;
59
60 for (i=0; i<n; i++){
61     cout <<"Enter Value and Weight: ";
62     cin >> Values[i] >> Weights[i];
63 }
64
65
66
67 cout << "Enter knapsack capacity: ";
68 int knapsack;
69 cin >> knapsack;
70
71 // for (i=0; i<12; i++){
72 float k = fractionknapsack(Values, Weights, knapsack, n) ;
73 cout << "\nMaximum Value: " << k;
74 //}
75
76 return 0;
77 }

```

5.2 Output



```

D:\College\DAA LAB\5.exe
Enter number of objects: 4
Enter Value and Weight: 1 1
Enter Value and Weight: 3 2
Enter Value and Weight: 5 3
Enter Value and Weight: 7 4
Enter knapsack capacity: 8

Maximum Value: 13.5
-----
Process exited after 18.44 seconds with return value 0
Press any key to continue . . .

```

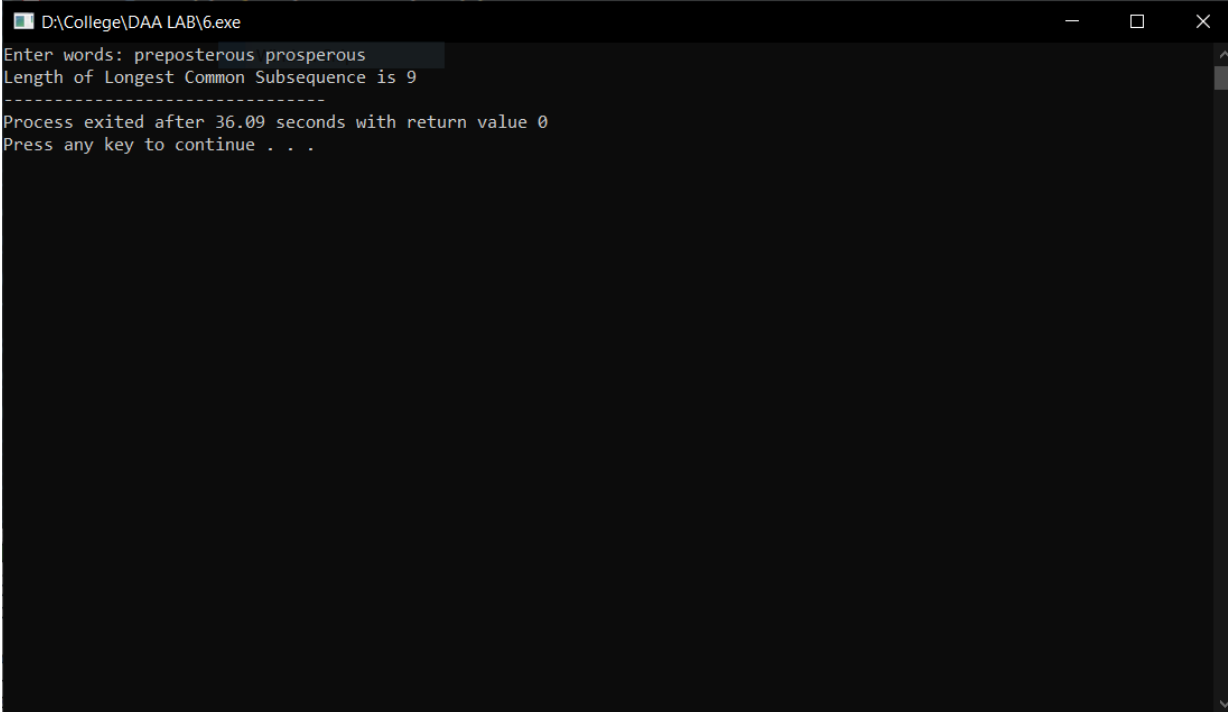
6 Longest Common Subsequence Algorithm

6.1 Question

6.2 Code

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int longestcommonsubseq(string a, string b){
6     int an = a.length(), bn = b.length();
7     int M[an+1][bn+1] = {0};
8
9
10    int i, j;
11
12
13    for (i=0; i<an+1; i++){
14        for (j=0; j<bn+1; j++){
15            M[i][j] = 0;
16        }
17    }
18    for (i=1; i<=an; i++){
19        for (j=1; j<=bn; j++){
20            M[i][j] = max(M[i][j-1], M[i-1][j]) + (a[i]==b[j]);
21            // cout << M[i][j]<<"\t ";
22        }
23        // cout <<"\n";
24    }
25
26    return M[an][bn];
27 }
28
29
30 int main(){
31     string a, b;
32
33     cout << "Enter words: ";
34
35     cin >> a >> b;
36
37     cout << "Length of Longest Common Subsequence is " << longestcommonsubseq(a, b);
38
39
40     return 0;
41 }
```

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

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Graph {
5     int v;
6     int e;
7     int** adj;
8
9     public:
10    Graph(int v, int e);
11    void addEdge(int start, int e);
12    void DFS(int start, vector<bool>& visited);
13 };
14
15 Graph::Graph(int v, int e)
16 {
17     this->v = v;
18     this->e = e;
19     adj = new int*[v];
20     for (int row = 0; row < v; row++) {
21         adj[row] = new int[v];
22         for (int column = 0; column < v; column++) {
23             adj[row][column] = 0;
24         }
25     }
26 }
27
28 void Graph::addEdge(int start, int e)
29 {
30     adj[start][e] = 1;
31     adj[e][start] = 1;
32 }
33
34 void Graph::DFS(int start, vector<bool>& visited)
35 {
36     cout << start << " ";
37     visited[start] = true;
38     for (int i = 0; i < v; i++) {
39         if (adj[start][i] == 1 && (!visited[i])) {
40             DFS(i, visited);
41         }
42     }
43 }
44
45 int main()
46 {
47     int v, e;
48     cout << "Enter number of nodes and edges: ";
49     cin >> v >> e;
50
51     Graph G(v, e);
52
53     int a, b;
54     for (int i = 0; i < e; i++) {
```

```

55         cout << "Enter nodes joined by edge: ";
56         cin>>a>>b;
57         G.addEdge(a,b);
58     }
59     vector<bool> visited(v, false);
60
61     int s;
62     cout << "Enter starting node: ";
63     cin>>s;
64
65     cout << "Nodes are visited as follows: ";
66
67     G.DFS(s, visited);
68 }

```

7.3 Output



```

C:\Users\LazyBoy\Downloads\dfs.exe
Enter number of nodes and edges: 6 6
Enter nodes joined by edge: 0 1
Enter nodes joined by edge: 0 2
Enter nodes joined by edge: 1 3
Enter nodes joined by edge: 1 4
Enter nodes joined by edge: 4 5
Enter nodes joined by edge: 5 2
Enter starting node: 0
Nodes are visited as follows: 0 1 3 4 5 2
-----
Process exited after 37.18 seconds with return value 0
Press any key to continue . . .

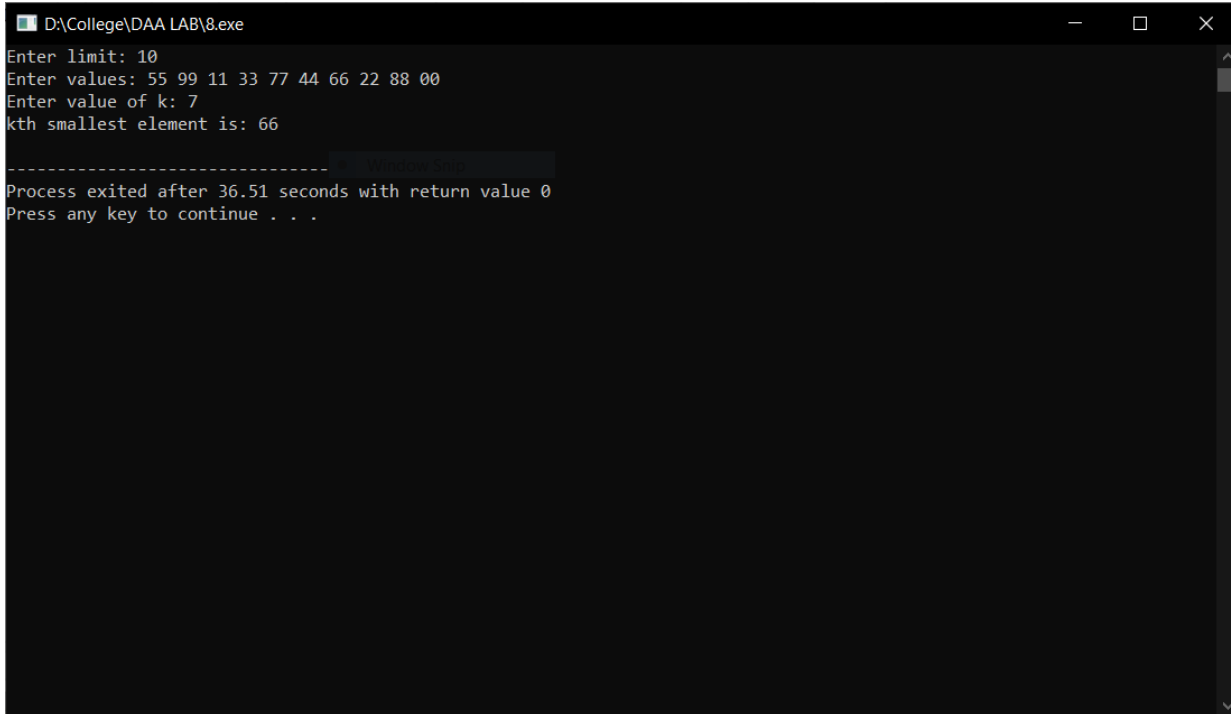
```

8 Finding Kth smallest element in worst case linear time

8.1 Code

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     int n,k;
7     cout<<"Enter limit: ";
8     cin>>n;
9     int arr[n];
10    cout<<"Enter values: ";
11    for(int i=0;i<n;i++)
12        cin>>arr[i];
13
14    cout<<"Enter value of k: ";
15    cin>>k;
16
17    int max = 0;
18    for(int i=0;i<n;i++)
19        if(arr[i]>max)
20            max=arr[i];
21
22    int count[max+1]={0};
23    for(int i=0;i<n;i++)
24        count[arr[i]]++;
25
26    int sum=0,res;
27    for(int i=0;i<=max;i++)
28    {
29        sum+=count[i];
30        if(sum>=k)
31        {
32            res=i;
33            break;
34        }
35    }
36    cout<<"kth smallest element is: "<<res<<endl;
37    return 0;
38 }
```

8.2 Output



A screenshot of a Windows command prompt window titled "D:\College\DAA LAB\8.exe". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The output text is as follows:

```
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 . . .
```

The text is displayed in a monospaced font on a black background. A vertical scrollbar is visible on the right side of the window.

9 Randomised Quick Sort

9.1 Code

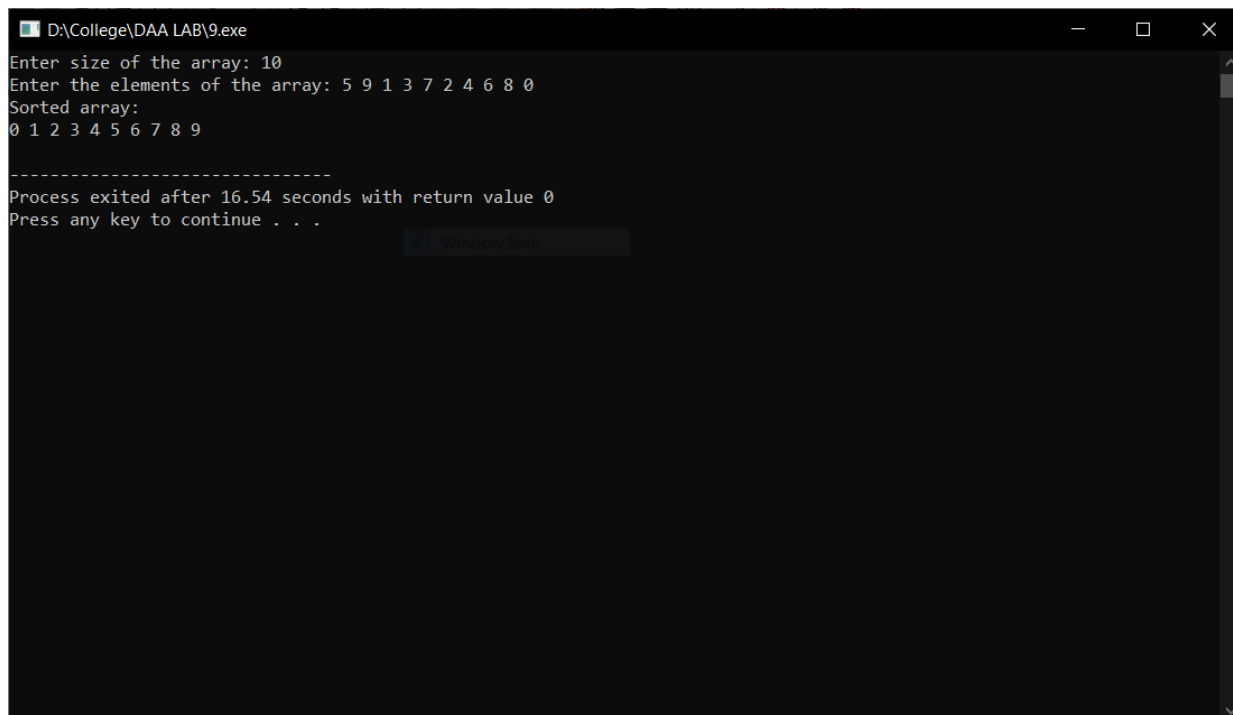
```
1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4
5
6 int partition(int arr[], int low, int high)
7 {
8     int pivot = arr[high];
9
10    int i = (low - 1);
11
12    for (int j = low; j <= high - 1; j++)
13    {
14
15        if (arr[j] <= pivot) {
16
17            i++;
18            swap(arr[i], arr[j]);
19        }
20    }
21    swap(arr[i + 1], arr[high]);
22    return (i + 1);
23 }
24
25
26 int partition_r(int arr[], int low, int high)
27 {
28     int random = low + rand() % (high - low);
29
30     swap(arr[random], arr[high]);
31
32     return partition(arr, low, high);
33 }
34
35
36 void quickSort(int arr[], int low, int high)
37 {
38     if (low < high) {
39
40         int pi = partition_r(arr, low, high);
41
42         quickSort(arr, low, pi - 1);
43         quickSort(arr, pi + 1, high);
44     }
45 }
46
47
48 void printArray(int arr[], int size)
49 {
50     int i;
51     for (i = 0; i < size; i++)
52         printf("%d ", arr[i]);
53     printf("\n");
54 }
55
56 int main()
```

```

57 {
58     int arr[20], i, n;
59     cout<<"Enter size of the array: ";
60     cin>>n;
61     cout<<"Enter the elements of the array: ";
62     for(i=0; i<n; i++)
63         cin>>arr[i];
64     quickSort(arr, 0, n - 1);
65     printf("Sorted array: \n");
66     printArray(arr, n);
67     return 0;
68 }

```

9.2 Output



```

D:\College\DAA LAB\9.exe
Enter size of the array: 10
Enter the elements of the array: 5 9 1 3 7 2 4 6 8 0
Sorted array:
0 1 2 3 4 5 6 7 8 9

-----
Process exited after 16.54 seconds with return value 0
Press any key to continue . . .

```

10 Convex Hull computation in 2D

10.1 Code

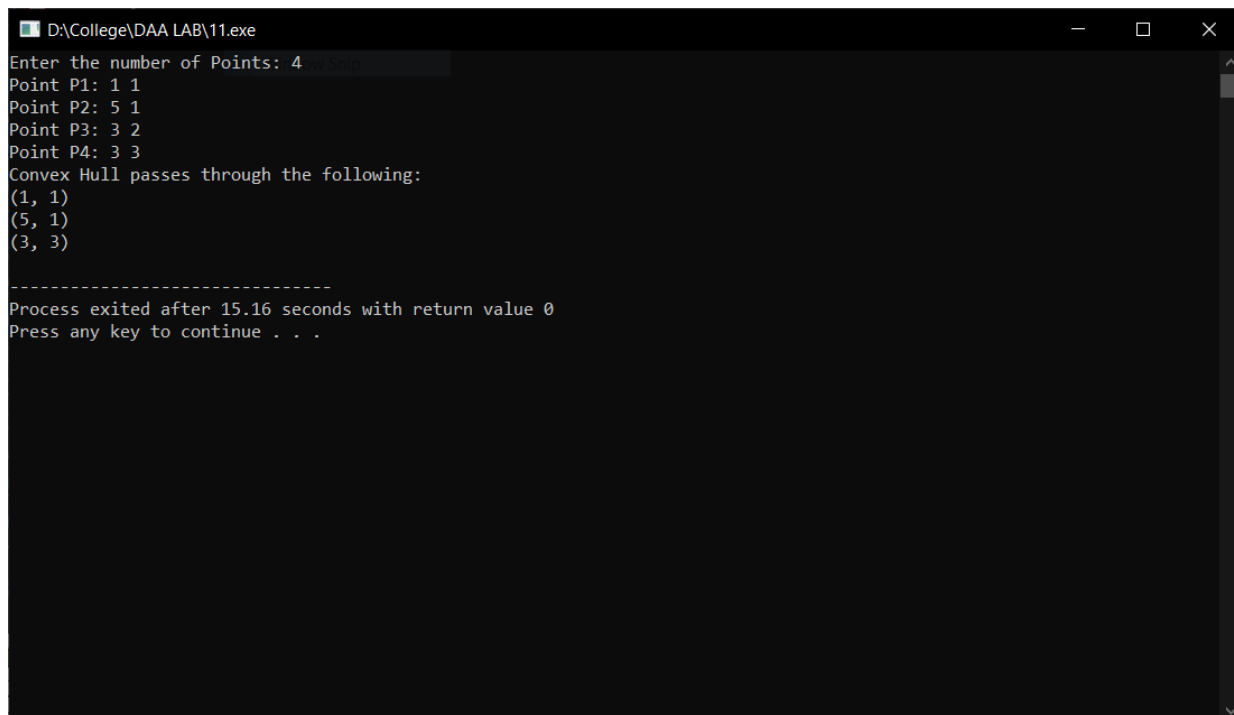
```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Point{
5     int x, y;
6 };
7
8 int orientation(Point p, Point q, Point r){
9     int val = (q.y - p.y) * (r.x - q.x) -
10             (q.x - p.x) * (r.y - q.y);
11
12     if (val == 0) return 0;
13     return (val > 0)? 1: 2;
14 }
15
16
17 void convexHull(Point points[], int n){
18     if (n < 3) return;
19
20     vector<Point> hull;
21
22     int l = 0;
23     for (int i = 1; i < n; i++){
24         if (points[i].x < points[l].x)
25             l = i;
26
27     int p = l, q;
28     do
29     {
30         hull.push_back(points[p]);
31         q = (p+1)%n;
32         for (int i = 0; i < n; i++){
33             if (orientation(points[p], points[i], points[q]) == 2)
34                 q = i;
35         }
36
37         p = q;
38     } while (p != l);
39
40     for (int i = 0; i < hull.size(); i++)
41         cout << "(" << hull[i].x << ", " << hull[i].y << ")\n";
42 }
43
44
45 int main(){
46     int num_points;
47     cout << "Enter the number of Points: ";
48     cin >> num_points;
49
50     if (num_points < 3)
51     {
52         cout << "\nConvex Hull is not possible\n";
53         return 0;
54     }
55
56 }
```

```

57     Point XY_plane[num_points];
58
59     for (int i = 0; i < num_points; i++)
60     {
61         printf("Point P%d: ", i + 1);
62         cin >> XY_plane[i].x >> XY_plane[i].y;
63     }
64
65     cout << "Convex Hull passes through the following: \n";
66     convexHull(XY_plane, num_points);
67     return 0;
68 }

```

10.2 Output



```

D:\College\DAA LAB\11.exe
Enter the number of Points: 4
Point P1: 1 1
Point P2: 5 1
Point P3: 3 2
Point P4: 3 3
Convex Hull passes through the following:
(1, 1)
(5, 1)
(3, 3)

-----
Process exited after 15.16 seconds with return value 0
Press any key to continue . . .

```