SHRI RAMSWAROOP MEMORIAL GROUP
OF PROFESSIONAL COLLEGES,
LUCKNOW

LAB WORK BOOK

For

DESIGN AND ANALYSIS OF ALGORITHM

LAB

(RCS-552)

Course: B.Tech.

Branch: CSE Year: 3rd Semester: 5th

Name of Student: Aayush Singh

College Roll No.: B617CS002 University Roll No.: 17122100002

Semester: 05 Session: 2019-20

Branch: Computer Science Group: CS-51

Lab Day: Tuesday Lab Period: 6, 7

Progress Sheet

S.No.	Experiment	Date Of Issue	Date Exp. Done	Date Exp. Checked	Marks Obtained	Authorized Signatory
1	SHEET 01	30/7	30/7	06/08	(5)	Am
2	SHEET 02	6/8	6/8	13/08	(4)	Am.
3	SHEET 03	13/8	13/8	20/08	(5)	Am
4	SHEET 04	20/8	20/8	27/08	(4)	Am
5	SHEET 05	27/8	27/8	24/09	(5)	Am
6	SHEET 06	24/9	24/9	24/09	(2.5)	Am
7	SHEET 07	24/9	24/9	01/10	(5)	Am
8	SHEET 08	1/10	1/10	01/10	(5)	Am
9	SHEET 09	22/10	22/10	22/10	(5)	Am
10	SHEET 10	22/10	22/10	22/10	(5)	Am

LAB - 1

Linear Search

```
import java.util.*;
class linear_search
{
public static void main (String [] args)
 {
 Scanner sc = new Scanner ( system.in );
 System.out.println ("Enter size of array");
 int n = sc.nextInt();
 int ar [] = new int [ar];
 system.out.println ("Enter array elements ");
 for ( int i=0; i<n; i++)
    ar [i] = sc.nextInt ();
 System.out.println ("The array is: ");
 for (int i=0; i<n ; i++)
    System.out.println ("ar [i] + "\t");
 System.out.println ("Enter element to be searched ");
 int x = sc.nextInt ();  int a, c = 0;
 for ( int i=0; i<n; i++)
  {
    if (ar [i] == x)
      { c++;
       break; a = i;
      }
    else
      continue;
    }
  if (c>0)
   system.out.println ("Element found & is at " + (a+1));
  else
   system.out.println ("Element not present ");
  }
}
```

LAB-2

Binary Search

```
import java.util.*;
class Binary_Search
{
public static void main (String [] args)
{
int i, fst, lst, mid, n, search, array[];
Scanner Sc= new Scanner (System.in);
System.out.println("Enter size of array");
n = Sc.nextInt();
array = new int [n];
System.out.println("Enter array elements");
for(i=0; i<n; i++)
  array = Sc.nextInt();
System.out.println("Enter element to be searched");
  search = Sc.nextInt();
  fst = 0;
  lst = n-1;
  mid = (fst+lst)/2;
  while (fst <= let).
{ if (array[mid] < search)
    fst = mid+1;
else if (array[mid] == search)
{ system.out.println(search + "found at "+(mid+1)+".");
  break;
}
else
  lst = mid-1;
  mid = (fst + lst)/2;
}
if (fst > lst)
System.out.println("Element not found");
 }
}
```

LAB - 3

Insertion Sort.

```
import java.util.*;
public class Insertion_sort
{
public static void main (String [] args)
{
Scanner sc = new Scanner (System.in);
System.out.println (" Enter Array size");
int n = sc.nextInt ();
system.out.println (" Enter Array Elements");
 int ar [] = new int [n];
 for (int i=0; i<n; i++)
     ar [i] = sc.nextInt ();
  Sort (ar);
  system.out.println ("Sorted Array");
  for (int i=0; i<n; i++)
      {
      system.out.println (ar [i] + "\t");
      }
  }
  public static void sort (int a[])
     {int x = a.length;
   for (int j=1; j< n; j++)
     {int key = a[j];
     int i = j-1;
     while ((i>-1) && (a[i]>key))
      { a [i+1] = a[i];
        i--;
      }
      a [i+1] = key;
     }
}
```

LAB - 4

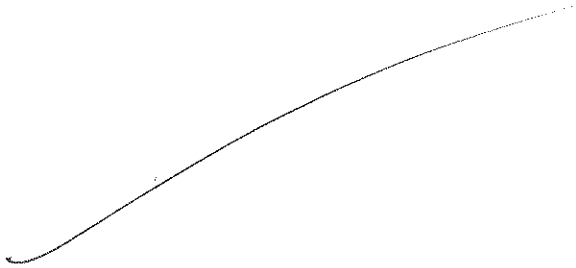
Merge Sort

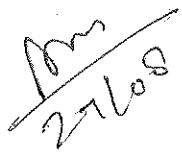
```
import java.util.*;
public class Merge_Sort
{
public static void main (String [] args)
{
Scanner sc = new Scanner (System.in);
System.out.println ("Enter array size");
int n = sc.nextInt();
int a[] = new int [n];
System.out.println ("Enter array elements");
for (int i=0; i<n; i++)
    a[i] = sc.nextInt();
    sort (a, 0, n-1);
System.out.println ("Sorted array");
for (int i=0; i<n; i++)
    System.out.println (a[i] + "\t");
}
public static void sort (int a[], int beg, int end)
{if (beg < end)
 {int mid = (beg + end)/2;
  sort (a, beg, mid);
  sort (a, mid+1, end);
  merge (a, beg, mid, end);
 }
public static void merge (int a[], int beg, int mid, int end)
{int l = mid - beg + 1;
 int r = end - mid;
 int left a[] = new int (l);
```

```
int right arr[] = new int[r];
for (int i=0; i<l; i++)
 left arr[i] = arr1[beg+i];
for (int j=0; j<r; j++)
right arr[j] = arr1[mid+1+j];
int i=0, j=0;
int k = beg;
while (i<l && j<r)
{ if (left arr[i] <= right arr[j])
  { arr1[k] = left arr[i];
      i++;
  }
else
  { arr1[k] = right arr[j];
      j++;
  } k++;
  }
while (i<l)
{ arr1[k] = left arr[i];
   i++;
   k++;
}
while (j<r)
{ arr1[k] = right arr[j];
      j++;
      k++;
}
}
```

Output:

```
Enter array size
10
Enter array elements
90
11
10
2
31
89
70
99
66
5
Sorted array
2 5 10 11 31 66 70 89 90 99
```





LAB - 05

Counting sort:-

```
import java.util.*;
class count_sort
{
public static void main (string args[])
{
Scanner sc = new scanner (system.in);
int ar[], n, max = 0
system.out.println("Enter array size");
n = sc.nextInt();
system.out.println("Enter array elements");
ar = newint[n];
for(int i=0; i<n; i++)
{ar[i] = sc.nextInt();
    if (max < ar[i])
        max = ar[i];
}
int c = newint[max]
for(int i=0; i<max; i++)
    c[i] = 0;
for(int i=0; i<max; i++)
    c[i] += c[i-1];
int b[] = new int[n];
for(int i=0; i<n; i++)
{ int t = [ar[i]] - 1;
        b[t] = ar[i];
        c[ar[i]] = c[ar[i]] - 1;
        }
for (int i=0; i<n; i++)
        system.out.println(ar[i]);
    }
}
```

O(n)

Lab - 6

Heap sort

```
public class Heapsort
{
public void sort(int arr[])
{
int n = arr.length;
for (int i= n/2-1 ; i>0; i--)
  heapify (arr, n, i)
for(int i = n-1; i>= 0; i--)
{ int temp = arr[0];
   arr[0] = arr[i];
   arr[i] = temp;
   heapify(arr, i, 0);
  }
 }
void heapify (int arr[], int n; int i)
{ int largest = i;
  int l = 2* i+1;
  int r = 2* i+2;
if (l < n && arr[i] > arr[length])
   largest = l;
if (r < n && arr[r] > arr[largest])
   largest = r;
if (largest != i)
  { int swap = arr[i];
    arr[i] = arr[largest];
    arr[largest] = swap;
    heapify (arr, n, largest);
  }
 }
static void printArray (int arr[])
```

```
{ int n = arr.length;
  for (int i=0; i<n; ++i)
System.out.println(arr[i]+" ");
System.out.println(); }

public static void main (String args[])
  {
   int arr[] = {12,11,13, 5, 6, 7};
   int n = arr.length;
   HeapSort ob = new HeapSort();
   ob.Sort(arr);
   System.out.println("Sorted array");
   printArray(arr);
  }
}
```

Output:

Sorted array

5 6 7 11 12 13

AM
24/09

Lab - 7

Implementation of knapsack problem by using greedy approach

```
import java.io.IOException
import java.util.Scanner
class fractional_knapsack
{public static void main (String args[]) throws IOException
 { int i, j=0, max qty, m, n;
   float sum=0, max;
  Scanner sc= new Scanner(System.in);
  int array[][] = new int [2][20];
  System.out.println("Enter no. of items");
   n= sc.nextInt();
  System.out.println("Enter weights of each time");
  for(i=0; i<n; i++)
    array[0][i]= sc.nextInt();
  System.out.println("Enter values of each item");
    for(i=0; i<n;i++)
    array[i][i]= sc.nextInt();
System.out.println("Enter max volume of knapsack");
 max_qty= sc.nextInt();
       m= max_qty;
  while(m>0)
  { max=0;
  for(i=0; i<n; i++)
  { if((float)array[i][i]/(float)array[0][i])>max)
     max= ((float)array[i][i]/((float)array[0][i];
        j=i;
     }}
    if(array[0][j]>m)
     {
  System.out.println("Quantity of item no."+(j+1)+"added is"+m);
    sum += m+max;
      m= -1; }
```

```
else
{ System.out.println("Quantity of item no." + "(j+1)" + "added is" +
                     array[0][j]);
  m = array[0][j];
  sum += (float) array[1][j];
     array[1][j] = 0;
  }}
```

O/P :-

```
Enter no. of items
5
Enter weight of each items
10  20  30  40  50
Enter values of each item
5   4   3   2   1
Enter maximum value of Knapsack:
80
Qty of item no.: 1 added is 10
Qty of item no.: 2 added is 20
Qty of item no.: 3 added is 30
Qty of item no.: 4 added is 20
The total profit is 130
```

AM
01/10

Lab - 8

Minimum spanning tree using Kruskal's algorithm

```
import java.util.*;
import java.lang.*;
import java.io.*;
class Graph
{
  class Edge implements Comparable<Edge>
  {
    int src, dest, weight;
    public int compareTo(Edge compareEdge)
    {
      return this.weight - compareEdge.weight;
    }
  };
class subset
{
  int parent, rank;
};
int V, E;
Edge edge[];
Graph(int v, int e)
{
  V = v;
  E = e;
  edge = new Edge[E];
  for(int i=0; i<e; ++i)
      edge[i] = new Edge();
  }
int find(subset subsets[], int i)
{
    if (subsets[i].parent != i)
      subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}
void Union(subset subsets[], int x, int y)
{
  int xroot = find(subsets, x);
```

```
int yroot = find (subsets, y);
if (subsets [xroot].rank < subsets [yroot].rank)
        subsets [xroot].parent = yroot;
else if (subsets[xroot].rank > subsets [yroot].rank)
        subsets [yroot].parent = xroot;

else
{ subsets [yroot].parent = xroot;
  subsets [xroot].rank++;
}
}
void KruskalMST()
{
    Edge result [] = new Edge [V];
    int e = 0;
    int i = 0;
    for ( i=0; i<V; ++i)
        result [i] = new Edge();
    Arrays. sort (edge);
    subset subsets [] = new subset [V];
    for (i=0; i<V; ++i)
        subsets [i] = new subset ();
    for (int v=0; v<V; ++v)
    { subsets [v].parent = v;
      subsets [v].rank = 0;
    }
    i = 0;
    while (e<V-1)
    {
      Edge next_edge = new Edge();
      next_Edge = edge [i++];
      int x = find ( subsets , next_edge. src);
      int y = find ( subsets ; next_edge. dest);
      if (x != y)
      { result [e++] = next_edge;
        Union (subsets, x, y);
      }
    }
    System. out. println ("Following are the edges in " + "the constructed MST");
```

```
for (i=0; i<e; ++i)
        System.out.println(result[i].src + "--" + result[i].dest + "==" + result[i].
        weight);
    }
public static void main (String[] args)
  { int V=4;
    int E=5;
    Graph graph = new Graph (V, E);
    graph.edge[0].src = 0;
    graph.edge[0].dest = 1;
    graph.edge[0].weight = 10;

    graph.edge[1].src = 0;
    graph.edge[1].dest = 2;
    graph.edge[1].weight = 6;
    graph.edge[2].src = 0;
    graph.edge[2].dest = 3;
    graph.edge[2].weight = 5;
    graph.edge[3].src = 1;
    graph.edge[3].dest = 3;
    graph.edge[3].weight = 15;
    graph.edge[4].src = 2;
    graph.edge[4].dest = 3;
    graph.edge[4].weight = 4;
      graph.KruskalMST();
  }
}
```

Am

Lab - 9

Travelling Salesman Problem Implementation

```
class GfG
{
  static int tsp (int[][] graph, boolean[]v, int currPos, int n,
    int count, int cost, int ans)
  {
    if (count == n && graph [currPos][0]>0)
    {
      ans = Math.min (ans, cost + graph [currPos][0]);
      return ans;
    }
    for (int i = 0; i<n; i++)
    {
      if (v[i] == false && graph [currPos[i]>0)
      {
        v[i] = true;
        ans = tsp (graph, v, i, n, count+1, cost +graph [currPos][i], ans);
        v[i] = false;
      }
    }
    return ans;
  }
public static void main (String [] args)
{ int n=4;
  int [][] graph = {{ 0, 10, 15, 20}, {10, 0, 35, 25}, { 15, 35, 0, 30}, {20, 25, 30, 0 }};
  boolean []v = new Boolean [n];
  v[0] = true;
  int ans = Integer.MAX-value;
  ans = tsp (graph, v, 0, n, 1, 0, ans);
  System.out.println (ans);
}
}
```

Am

Lab - 10

```
#N - Queen Problem implementation
public class NQueenProblem {
final int N = 4;
 void printSolution ( int board [][])
 {
  for (int i=0; i<N; i++) {
    for (int j =0; j<N; j++)
        System. out. print (" " + board [i][j] + " ");
     System .out.println ();
    }
 }
boolean isSafe (int board[][], int row, int col)
{ int i, j;
  for (i=0; i < col; i++)
     if (board [row][i] == 1)
        return false;
  for (i=row, j =col ; i >= 0 && j >= 0; i--; j--)
      if ( board [i][j] == 1)
         return false ;
  for (i =row, j= col; j>=0 && i < N; i++; j--)
     if (board [i] [j] = = 1)
        return false;
  return true; }
boolean SolveNQUtil (int board [][], int col)
{ if (col >= N)
   return true;
 for (int i=0; i<N; i++){
     if (isSafe (board, i, col)){
       board [i] [col ] = 1;
      if (solveNQUtil (board, col, +1) == true )
         return true;
       board [i] [col] = 0;
      }
   }
   return false;
}
```

```
boolean  solveNQ()
{
  int board[][]= {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
  if (solveNQUtil(board,0)==false){
      System.out.print("Solution does not exist");
      return false;
    }
  printSolution(board);
  return true;
}
public static void main(String args[])
  {
    NQueenProblem Queen = new NQueenProblem();
    Queen.solveNQ();
  }
}
```

Am