Course Code : BCSL-045 Course Title : Introduction to Algorithm Design Lab Assignment Number : BCA(4)/L-045/Assignment/16-17 Maximum Marks : 50

1. Write a programe to perform matrix multiplication of 3 x 3 using Strassen's algorithm. Calculate total number of addition and multiplication operations while computing the matrix multiplication. (6 marks)

ANS 1 #include<stdio.h>

```
int main() {
  int a[10][10], b[10][10], c[10][10], i, j, k;
  int sum = 0;

  printf("\nEnter First Matrix : n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      scanf("%d", &a[i][j]);
    }
  }

  printf("\nEnter Second Matrix:n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
```

```
      scanf("%d", &b[i][j]);
    }
  }

  printf("The First Matrix is: \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", a[i][j]);
    }
    printf("\n");
  }

  printf("The Second Matrix is : \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", b[i][j]);
    }
    printf("\n");
  }

  //Multiplication Logic
  for (i = 0; i <= 2; i++) {
    for (j = 0; j <= 2; j++) {
      sum = 0;
      for (k = 0; k <= 2; k++) {
        sum = sum + a[i][k] * b[k][j];
      }
      c[i][j] = sum;
    }
  }

  printf("\nMultiplication Of Two Matrices : \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", c[i][j]);
    }
    printf("\n");
  }
```

```
   return (0);
}
```

2. Implement a Bubble Sort algorithm for sorting the following list of numbers and showing the list obtained at each step. 17 10 7 9 25 30 8 Calculate the total number of exchange operations. How many times will the loop execute ? (5 marks)

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

int binarySearch(const int a[], int size, int key);
int binarySearch(const int a[], int iLeft, int iRight, int key);
void print(const int a[], int iLeft, int iRight);

int main() {
   const int SIZE = 10;
   int a1[SIZE] = {1, 4, 5, 8, 12, 19, 24, 31, 43, 55}; // sorted

   cout << binarySearch(a1, SIZE, 8) << endl;
   cout << binarySearch(a1, SIZE, 12) << endl;
   cout << binarySearch(a1, SIZE, 24) << endl;
   cout << binarySearch(a1, SIZE, 21) << endl;
}

// Search the array for the given key
// If found, return array index; otherwise, return -1
int binarySearch(const int a[], int size, int key) {
```

```
   // Call recursive helper function
   return binarySearch(a, 0, size-1, key);
}

// Recursive helper function for binarySearch
int binarySearch(const int a[], int iLeft, int iRight, int key) {
   // For tracing the algorithm
   print(a, iLeft, iRight);

   // Test for empty list
   if (iLeft > iRight) return -1;

   // Compare with middle element
   int mid = (iRight + iLeft) / 2;  // truncate
   if (key == a[mid]) {
      return mid;
   } else if (key < a[mid]) {
      // Recursively search the lower half
      binarySearch(a, iLeft, mid - 1, key);
   } else {
      // Recursively search the upper half
      binarySearch(a, mid + 1, iRight, key);
   }
}

// Print the contents of the given array from iLeft to iRight (inclusive)
void print(const int a[], int iLeft, int iRight) {
   cout << "{";
   for (int i = iLeft; i <= iRight; ++i) {
      cout << a[i];
      if (i < iRight) cout << ",";
   }
   cout << "}" << endl;
}
```

3. Write a programe to implement a connected graph (all vertices are connected to each other) with five vertices in form of an

adjancy list and then implement DFS to find a Spanning Tree of the graph. (6 marks)

```
ANS 3 #include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
 int i;
 reach[v]=1;
 for(i=1;i<=n;i++)
  if(a[v][i] && !reach[i])
  {
   printf("\n %d->%d",v,i);
   dfs(i);
  }
}
void main()
{
 int i,j,count=0;
 clrscr();
 printf("\n Enter number of
vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  reach[i]=0;
  for(j=1;j<=n;j++)
```

```
    a[i][j]=0;
  }
  printf("\n Enter the adjacency
matrix:\n");
  for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
    scanf("%d",&a[i][j]);
  dfs(1);
  printf("\n");
  for(i=1;i<=n;i++)
  {
   if(reach[i])
    count++;
  }
  if(count==n)
   printf("\n Graph is connected");
  else
   printf("\n Graph is not
connected");
  getch();
```

4. Given an ordered list of n integers and integer x. Find the number of comparisons used to determine the position of an integer x in the list using a Liner Search Algorithm. (6 marks)

ANS 4 `#include<iostream>`

```
using namespace std;

// Simple binary search algorithm
int binarySearch(int arr[], int l, int r, int x)
{
    if (r>=l)
    {
        int mid = l + (r - l)/2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid-1, x);
        return binarySearch(arr, mid+1, r, x);
    }
    return -1;
}

// function takes an infinite size array and a key to be
//  searched and returns its position if found else -1.
// We don't know size of arr[] and we can assume size to be
// infinite in this function.
// NOTE THAT THIS FUNCTION ASSUMES arr[] TO BE OF INFINITE SIZE
// THEREFORE, THERE IS NO INDEX OUT OF BOUND CHECKING
int findPos(int arr[], int key)
{
    int l = 0, h = 1;
    int val = arr[0];

    // Find h to do binary search
    while (val < key)
    {
        l = h;         // store previous high
        h = 2*h;       // double high index
        val = arr[h]; // update new val
    }

    // at this point we have updated low and high indices,
    // thus use binary search between them
    return binarySearch(arr, l, h, key);
}

// Driver program
int main()
{
    int arr[] = {3, 5, 7, 9, 10, 90, 100, 130, 140, 160, 170};
    int ans = findPos(arr, 10);
    if (ans==-1)
        cout << "Element not found";
    else
```

```
        cout << "Element found at index " << ans;
    return 0;
}
```

5. Write a program to check whether a string is a palindrome or not and calculate (a) Total number of swap operations (b) Number of times the loop will execute. (6 marks)

```
#include <stdio.h>
#include <string.h>
int main(){
char string1[20];
int i, length;
int flag = 0;
printf("Enter a string:");
scanf("%s", string1);
length = strlen(string1);
for(i=0;i < length ;i++)
{ if(string1[i] != string1
[length-i-1]){ flag = 1; break; }
} if (flag)
{ printf("%s is not a palindrome", string1); }
 else
```

```
{ printf("%s is a palindrome", string1);
} return 0;
}
```

6. Write a programe to locate the last occurrence of the smallest element in a finite list of integers. How many comparison operations will occur ? (5 marks)

ANS 6

procedure lastsmallest

(a_1,a_2,...,a_n: integers)

min = a_1 location = 1

for i = 2 to n if min >= a_

i then min = a

_i location = i

end

7. Implement Merge Sort algorithm to sort the following array 85 95 35 102 15 60 70 25 9

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

    // Driver method
    public static void main(String args[])
    {
        int arr[] = {85 ,95, 35, 102, 15 ,60, 70 ,25,9};

        System.out.println("Given Array");
        printArray(arr);

        MergeSort ob = new MergeSort();
        ob.sort(arr, 0, arr.length-1);

        System.out.println("\nSorted array");
        printArray(arr);
    }
}
```