DESIGN AND ANALYSIS OF ALGORITHMS

LAB MANUAL

YEAR	**:**	**2022-2023**
COURSE CODE	**:**	**CA-C19L**
SEMESTER	**:**	**IV**
COURSE	**:**	**BCA**

ACHARYA INSTITUTE OF GRADUATE STUDIES

(NAAC Reaccredited 'A' Grade and Affiliated to Bengaluru City University)

1#89/90, Soldevanahalli, Hesaraghatta road, BENGALURU – 560107

1.SYLLABUS:

ANALYSIS AND DESIGN OF ALGORITHMS LAB				
III Semester BCA				
Course Code	**Category**	**Hours / Week**	**Credits**	**Maximum Marks**
CA-C19L				
	Tutorial classes : Nil	Practical Classes: 39		Total Classes: 39

OBJECTIVES:

The course should enable the students to:

Learn how to analyse a problem and design the solution for the problem.

I. Design and implement efficient algorithms for a specified application.

II. Strengthen the ability to identify and apply the suitable algorithm for the given real world problem.

LIST OF EXPERIMENTS	
WEEK-1	**LINEAR SEARCH**
1. Write a program to implement linear search algorithm Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.	
WEEK-2	**BINARY SEARCH**
2. Write a program to implement binary search algorithm. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.	
WEEK-3	**TOWER OF HONAI**
3.Write a program to solve towers of honai problem and execute it for different number of disks	
WEEK-4	**SELECTION SORT**
4. Write a Program to Sort a given set of numbers using selection sort algorithm. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.	
WEEK-5	**BRUTE FORCE DIVIDE AND CONQUER**
5. Write a program to find the value of an (where a and n are integers) using both brute-force based algorithm and divide and conquer based algorithm	
WEEK-6	**QUICK SORT**

6. Write a Program to Sort a given set of elements using quick sort algorithm. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.	
WEEK-7	**BINOMIAL COEFFICIENT**
7. Write a Program to find the binomial co-efficient C(n, k), [where n and k are integers and n > k] using brute force based algorithm and also dynamic programming based algorithm	
WEEK-8	**FLOYD'S ALGORITHM**
8. Write a Program to implement Floyd's algorithm and find the lengths of the shortest paths from every pairs of vertices in a given weighted graph	
WEEK-9	**POLYNOMIAL HORNERS RULE**
9. Write a program to evaluate a polynomial using brute-force based algorithm and using Horner's rule and compare their performances	
WEEK-10	**STRING MATCHING BOYER MOORE**
10. Write a Program to solve the string matching problem using Boyer-Moore approach.	
WEEK-11	**STRING MATCHING KMP ALGORITHM**
11. Write a Program to solve the string matching problem using KMP algorithm	
WEEK-12	**BFS**
12. Write a program to implement BFS traversal algorithm	
WEEK-13	**PRIM'S Algorithm**
13. Write a program to find the minimum spanning tree of a given graph using Prim's algorithm	
WEEK-14	**WARSHALL'S Algorithm**
14. Write a Program to obtain the topological ordering of vertices in a given digraph. Compute the transitive closure of a given directed graph using Warshall's algorithm.	
WEEK-15	**SUM OF SUBSETS**
15. Write a Program to Find a subset of a given set S = {s1,s2, ,sn} of n positive integers whose ..... sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.	

Reference Books:

1. Levitin A, “Introduction to the Design And Analysis of Algorithms”, Pearson Education, 2008.

2. Goodrich M.T.,R Tomassia, “Algorithm Design foundations Analysis and Internet Examples”, John Wileyn and Sons, 2006.

3. Base Sara, Allen Van Gelder ,“ Computer Algorithms Introduction to Design and Analysis”, Pearson, 3rd Edition, 1999.

Web References:

1. **https://www.geeksforgeeks.org/design-and-analysis-of-algorithms/**
2. **https://www.javatpoint.com/daa-tutorial**
3. **https://www.codingninjas.com/codestudio/library/design-and-algorithm-analysis**

SOFTWARE AND HARDWARE REQUIREMENTS FOR A BATCH OF 36 STUDENTS:

HARDWARE:

Desktop Computer Systems: 36 nos

SOFTWARE:

Application Software: C Programming Compiler

S.NO	EXPERIMENT	DATES
1	LINEAR SEARCH	
2	BINARY SEARCH	
3	TOWER OF HONAI	
4	SELECTION SORT	
5	BRUTE FORCE DIVIDE AND CONQUER	
6	QUICK SORT	
7	BINOMIAL COEFFICIENT	
8	FLOYD'S ALGORITHM	
9	POLYNOMIAL HORNERS RULE	
10	STRING MATCHING BOYER MOORE	
11	STRING MATCHING KMP ALGORITHM	
12	BFS	
13	PRIM'S Algorithm	
14	WARSHALL'S Algorithm	
15	SUM OF SUBSETS	

WEEK -1

Objective:

To implement linear search algorithm Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.

PROCEDURE:

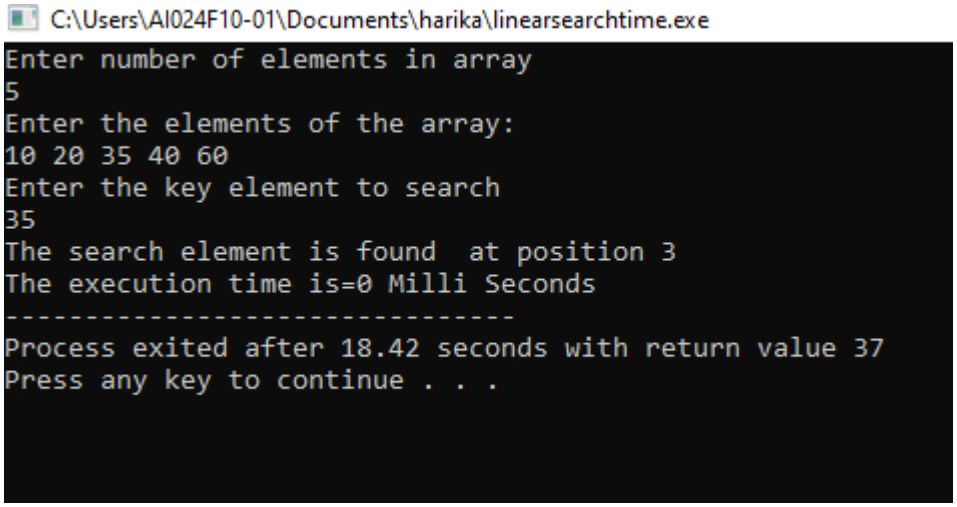
1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include <stdio.h>
#include<time.h>
#include<stdlib.h>
int linearSearch(int a[],int n,int key)
{
int i;
for(i=0;i<n;i++)
{
if(key==a[i])
return i;
}
return -1;
}
void main()
{
 char ch;
 int a[100],n,key,res,i;
 clock_t st,et;

  printf("Enter number of elements in array \n");
  scanf("%d", &n);
  printf("Enter the elements of the array: \n");
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  printf("Enter the key element to search\n");
  scanf("%d", &key);

st=clock();
res=linearSearch(a,n,key);
et=clock();
double time_taken=(((double)(et-st))/CLOCKS_PER_SEC)*1000;

if(res==-1)
```

```
{
    printf("The search element is not found \n");
    printf("The execution time is=%.0f Milli Seconds",time_taken);
exit(0);

}
else
printf("The search element is found  at position %d\n",res+1);
printf("The execution time is=%.0f Milli Seconds",time_taken);

}
```

OUTPUT:



WEEK -2

Objective:

To implement binary search algorithm. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

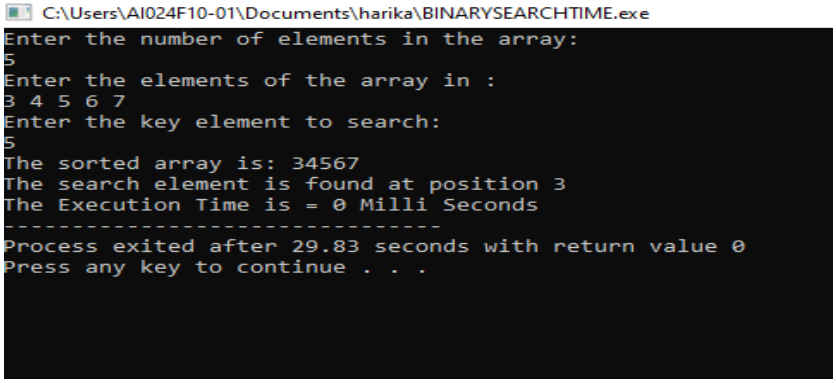
SOURCE CODE:

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>

int binarySearch(int a[],int key,int n,int first,int last)

{

int mid,i,j,temp;
```

```
if(last<first)
 return -1;
for(i=0;i<=n-2;i++)
{
for(j=0;j<=n-2;j++)
{
if(a[j+1]<a[j])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
while(first<=last)
{
mid=(first+last)/2;
if(key==a[mid])
    return mid+1;
else if(key<a[mid])
      last=mid-1;
else
     first=mid+1;
}
 return -1;
}
int main()
{
char ch;
int a[100],n, key, i, res, first, last;
clock_t st,et;
printf("Enter the number of elements in the array: \n");
scanf("%d",&n);
printf("Enter the elements of the array in :\n");
```

```
 for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter the key element to search: \n");
 scanf("%d",&key);
first=0;
last=n-1;
st=clock();
// Record Start Time
res=binarySearch(a,key,n,first,last);
printf("The sorted array is: ");
for(i=0;i<n;i++)
printf("%d",a[i]);
et=clock();
// Record End time
double time_taken = (((double) (et - st)) / CLOCKS_PER_SEC)*1000;
if(res ==-1)
{
printf("\nThe search element is not found\n");
printf("The Execution Time is = %.0f Milli Seconds",time_taken);
exit(0);
}
else
      printf("\nThe search element is found at position %d\n",res);
printf("The Execution Time is = %.0f Milli Seconds",time_taken);
}
```

OUTPUT:



WEEK -3
Objective:

To solve towers of honai problem and execute it for different number of disks

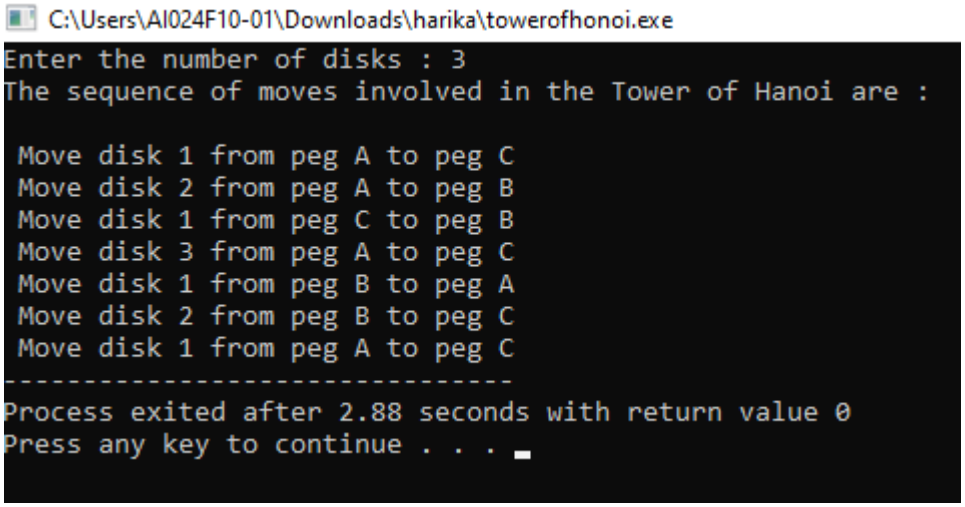
PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include <stdio.h>

void towers(int, char, char, char);

int main()
{
   int num;
   printf("Enter the number of disks : ");
   scanf("%d", &num);
   printf("The sequence of moves involved in the Tower of Hanoi are :\n");
   towers(num, 'A', 'C', 'B');
   return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
   // Base Condition if no of disks are
   if (num == 1)
   {
      printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
      return;
   }

   // Recursively calling function twice
   towers(num - 1, frompeg, auxpeg, topeg);
   printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
   towers(num - 1, auxpeg, topeg, frompeg);
  }
```

OUTPUT:



WEEK -4

Objective:

To Sort a given set of numbers using selection sort algorithm. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

PROCEDURE:

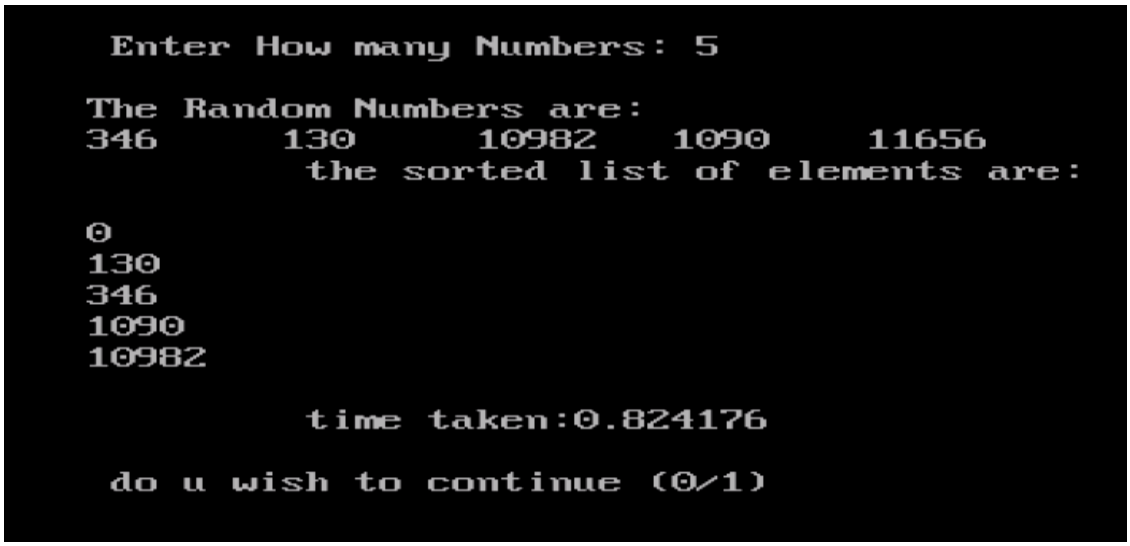
1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
void main()
{
int i,n,j,min,k,a[20],ch=1;
clock_t begin,end;
while(ch)
{
```

```
printf("\n Enter How many Numbers: ");
scanf("%d", &n);
printf("\nThe Random Numbers are:\n");
for(k=1; k<=n; k++)
{
a[k]=rand();
printf("%d\t",a[k]);
}
begin=clock();
for(k=0;k<=n-2;k++)
{
min=k;
delay(200);
for(j=k+1;j<=n;j++)
{
  if(a[j]<a[min])
          min=j;
}
i=a[k];
a[k]=a[min];
a[min]=i;
}
end=clock();
printf("\n\t the sorted list of elements are:\n");
for(k=0;k<n;k++)
printf("\n%d",a[k]);
printf("\n\n\t time taken:%lf",(end-begin)/CLK_TCK);
printf("\n\n do u wish to continue (0/1)\n");
scanf("%d",&ch);
}
getch();
}
```

OUTPUT:



WEEK -5

Objective:

To find the value of an (where a and n are integers) using both brute-force based algorithm and divide and conquer based algorithm.

PROCEDURE:

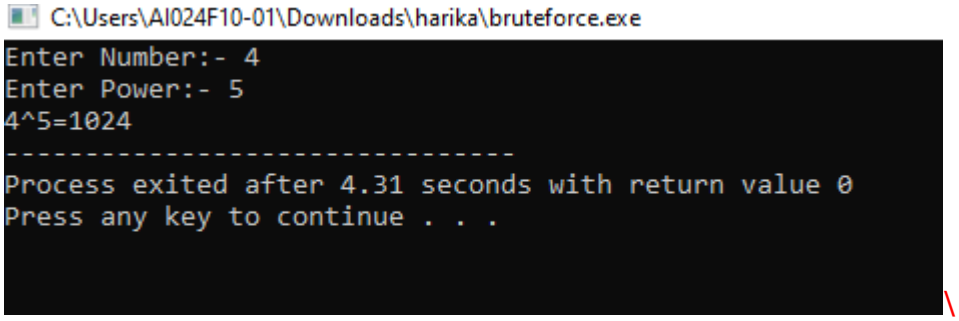
1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
long int power(int x,int n)
{
if(n==0)
{
return 1;
}
else if(n%2==0)
{
return power(x,n/2)*power(x,n/2);
}
else
{
return x*power(x,(n-1)/2)*power(x,(n-1)/2);
}
}
```

```
int main()
{
int x,y;
printf("Enter Number:- ");
scanf("%d",&x);
printf("Enter Power:- ");
scanf("%d",&y);
printf("%d^%d=%d",x,y,power(x,y));
return 0;
}
```

OUTPUT:



\

WEEK -6

Objective:

To Sort a given set of elements using quick sort algorithm. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include <stdio.h>
#include <conio.h>
#include <time.h>
void quicksort(int A[10], int low, int high)
{
int i,j,pivot,temp;
if (low < high)
{
pivot=low;
   i=low;
```

```
   j=high;
while (i<j)
{
while(A[i]<=A[pivot]&&i<high)
i++;
while (A[j]>A[pivot])
j--;
if (i<j)
{
temp = A[i];
A[i] = A[j];
A[j] = temp;
}
}
temp = A[pivot];
A[pivot] = A[j];
A[j] = temp;
quicksort(A,low,j-1);
quicksort(A,j+1,high);
}
}
int main()
{
int i, n, A[10];
clock_t st, et;
printf("Enter the number of elements of array: \n");
scanf("%d", &n);
printf("Enter the elements of the array: \n");
for (i=0; i< n; i++)
scanf(" %d", &A[i]);
st = clock();
quicksort(A, 0, n-1);
et = clock();
double time_taken = (((double) (et - st)) / CLOCKS_PER_SEC)*1000;
printf("Sorted list of elements:");
for (i=0; i<n; i++)
printf(" %d ", A[i]);
printf("The Execution Time is = %.0f Milli Seconds",time_taken);
```

```
return 0;
}
```

OUTPUT:

```
Enter the number of elements of array:
5
Enter the elements of the array:
3 2 1 6 9
Sorted list of elements: 1  2  3  6  9 The Execution Time is = 0 Milli Seconds
--------------------------------
Process exited after 9.496 seconds with return value 0
Press any key to continue . . .
```

WEEK -7

Objective:

To find the binomial co-efficient C(n, k), [where n and k are integers and n > k]

using brute force based algorithm and also dynamic programming based algorithm.

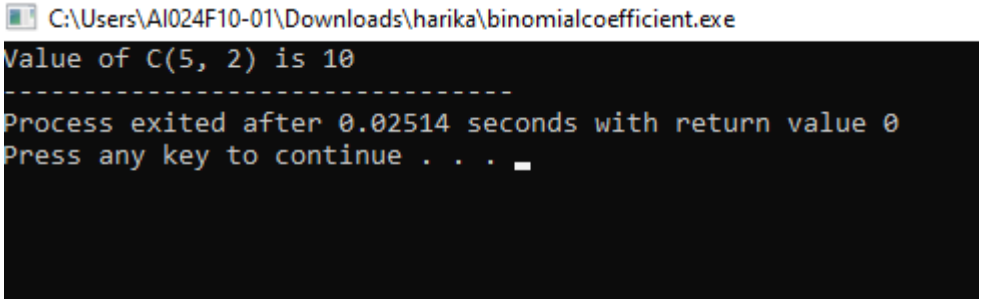
PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include <stdio.h>
 // Returns value of Binomial Coefficient C(n, k)
int binomialCoeff(int n, int k)
{
   // Base Cases
   if (k > n)
      return 0;
   if (k == 0 || k == n)
      return 1;

   // Recur
   return binomialCoeff(n - 1, k - 1)
         + binomialCoeff(n - 1, k);
}
 /* Driver program to test above function*/
int main()
{
   int n = 5, k = 2;
   printf("Value of C(%d, %d) is %d ", n, k,
   binomialCoeff(n, k));
   return 0;
}
```

OUTPUT:



WEEK -8

Objective:

To implement Floyd's algorithm and find the lengths of the shortest paths from every pairs of vertices in a given weighted graph.

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int cost[10][10],a[10][10];
void all_paths(int [10][10],int [10][10],int);
int min1(int,int);
int main()
{
        int i,j,n;
        printf("\n enter the number of vertices\n");
        scanf("%d",&n);
        printf("\n enter the adjacency matrix\n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&cost[i][j]);
        all_paths(cost,a,n);
        printf("\n\t the shortest path obtained is\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
```

```
            printf("\t %d",a[i][j]);
            printf("\n");
        }
        return 0;
}
void all_paths(int cost[10][10],int a[10][10],int n)
{
        int i,j,k;
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        a[i][j]=cost[i][j];
        for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        a[i][j]=min1(a[i][j],a[i][k]+a[k][j]);
}
int min1(int a,int b)
{
        return(a<b)?a:b;
}
```

OUTPUT:

```
 enter the number of vertices
4

 enter the adjacency matrix
999 999 3 999
2 999 999 999
999 7 999 1
6 999 999 999

         the shortest path obtained is
         10      10      3       4
         2       12      5       6
         7       7       10      1
         6       16      9       10

--------------------------------
Process exited after 33.26 seconds with return value 0
Press any key to continue . . .
```

WEEK -9

Objective:

To evaluate a polynomial using brute-force based algorithm and using Horner's rule and compare their performances

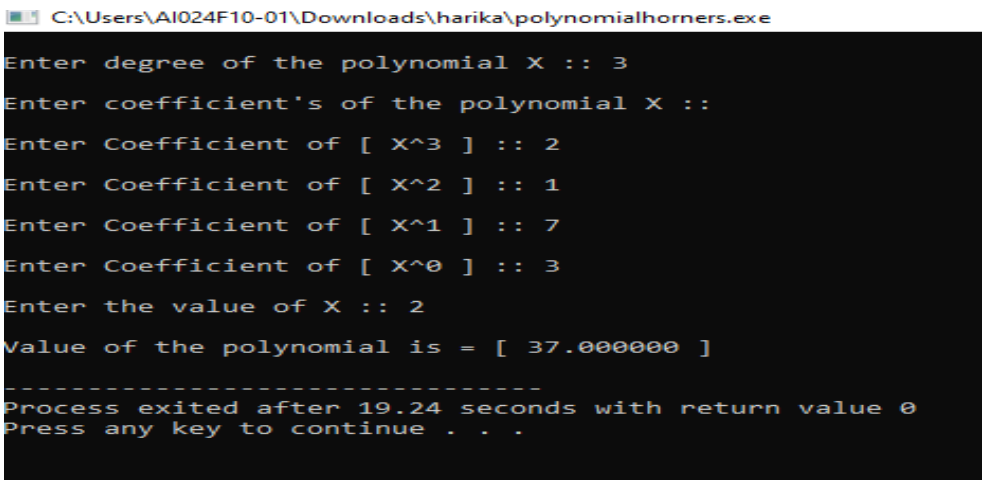
PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include <stdio.h>
int main()
{
    float a[100],sum=0,x;
    int n,i;

    printf("\nEnter degree of the polynomial X :: ");
    scanf("%d",&n);
    printf("\nEnter coefficient's of the polynomial X :: \n");
    for(i=n;i>=0;i--)
    {
        printf("\nEnter Coefficient of [ X^%d ] :: ",i);
        scanf("%f",&a[i]);
    }
        printf("\nEnter the value of X :: ");
        scanf("%f",&x);

    for(i=n;i>0;i--)
    {
      sum=(sum+a[i])*x;
    }
sum=sum+a[0];
printf("\nValue of the polynomial is = [ %f ]\n",sum);
  return 0;
}
```

OUTPUT:



WEEK -10

Objective:

To solve the string matching problem using Boyer-Moore approach.

PROCEDURE:

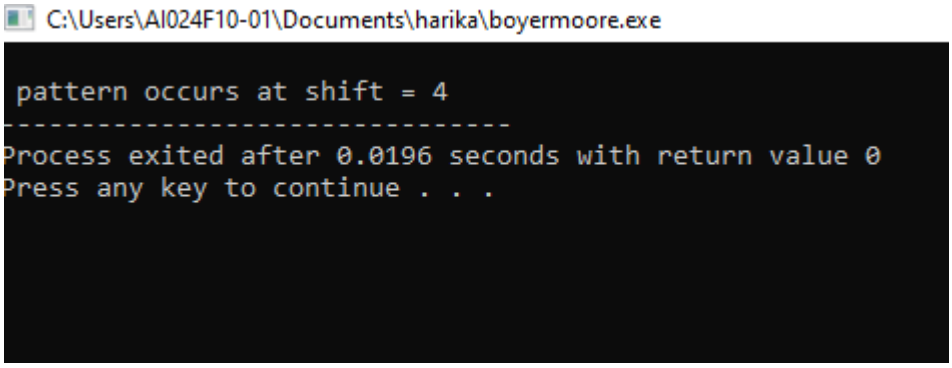
1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
# include <limits.h>
# include <string.h>
# include <stdio.h>

# define NO_OF_CHARS 256

// A utility function to get maximum of two integers
int max(int a, int b) {
    return (a > b) ? a : b;
}

// The preprocessing function for Boyer Moore's bad character heuristic
void badCharHeuristic(char *str, int size, int badchar[NO_OF_CHARS])
{
    int i;

    // Initialize all occurrences as -1
    for (i = 0; i < NO_OF_CHARS; i++)
        badchar[i] = -1;

    // Fill the actual value of last occurrence of a character
    for (i = 0; i < size; i++)
        badchar[(int) str[i]] = i;
}

void search(char *txt, char *pat) {
    int m = strlen(pat);
    int n = strlen(txt);

    int badchar[NO_OF_CHARS];

    badCharHeuristic(pat, m, badchar);

    int s = 0; // s is shift of the pattern with respect to text
    while (s <= (n - m)) {
        int j = m - 1;
```

```
        while (j >= 0 && pat[j] == txt[s + j])
            j--;

        if (j < 0) {
            printf("\n pattern occurs at shift = %d", s);

            s += (s + m < n) ? m - badchar[txt[s + m]] : 1;

        }

        else
            s += max(1, j - badchar[txt[s + j]]);
    }
}

int main() {
    char txt[] = "ABAAABCD";
    char pat[] = "ABC";
    search(txt, pat);
    return 0;
}
```

OUTPUT:



WEEK -11

Objective:

To solve the string matching problem using KMP algorithm

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

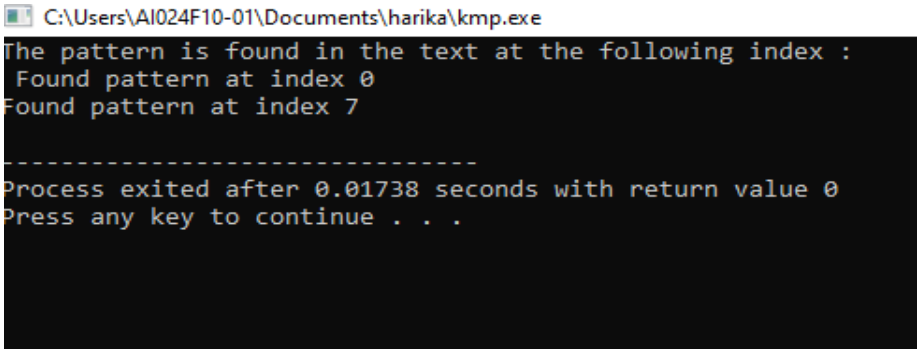
SOURCE CODE:

```
#include<stdio.h>
```

```
#include<string.h>
void prefixSuffixArray(char* pat, int M, int* pps)
{
  int length = 0;
  pps[0] = 0;
  int i = 1;
  while (i < M) {
    if (pat[i] == pat[length]) {
      length++;
      pps[i] = length;
      i++;
    } else {
      if (length != 0)
      length = pps[length - 1];
      else {
        pps[i] = 0;
        i++;
      }
    }
  }
}
void KMPAlgorithm(char* text, char* pattern) {
  int M = strlen(pattern);
  int N = strlen(text);
  int pps[M];
  prefixSuffixArray(pattern, M, pps);
  int i = 0;
  int j = 0;
  while (i < N) {
    if (pattern[j] == text[i]) {
      j++;
      i++;
    }
    if (j == M) {
      printf("Found pattern at index %d \n", i - j);
      j = pps[j - 1];
    }
    else if (i < N && pattern[j] != text[i]) {
      if (j != 0)
      j = pps[j - 1];
      else
      i = i + 1;
  }
}
```

```
}
int main()
{
  char text[] = "xyztrwqxyzfg";
  char pattern[] = "xyz";
  printf("The pattern is found in the text at the following index : \n ");
  KMPAlgorithm(text, pattern);
  return 0;
}
```

OUTPUT:



WEEK -12

Objective:

To implement BFS traversal algorithm

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.

2. Compile: Alt + F9

3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v)
{
        for (i=1;i<=n;i++)
          if(a[v][i] && !visited[i])
           q[++r]=i;
        if(f<=r) {
                visited[q[f]]=1;
                bfs(q[f++]);0
        }
}
int main()
```

```
{
        int v;
        printf("\n Enter the number of vertices:");
        scanf("%d",&n);
        for (i=1;i<=n;i++)
         {
                q[i]=0;
                visited[i]=0;
        }
        printf("\n Enter graph data in matrix form:\n");
        for (i=1;i<=n;i++)
          for (j=1;j<=n;j++)
           scanf("%d",&a[i][j]);
        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
        bfs(v);
        printf("\n The node which are reachable are:\n");
        for (i=1;i<=n;i++)
          if(visited[i])
           printf("%d\t",i);

          else
           printf("\n Bfs is not possible");
        getch();
}
```

OUTPUT:

```
 Enter the number of vertices:5

 Enter graph data in matrix form:
0 1 1 0 0
1 0 1 1 1
1 1 0 1 0
0 1 1 0 1
0 1 0 1 0

 Enter the starting vertex:1

 The node which are reachable are:
1       2       3       4       5
--------------------------------
Process exited after 83.66 seconds with return value 0
Press any key to continue . . .
```

WEEK -13

Objective:

To find the minimum spanning tree of a given graph using Prim's algorithm

PROCEDURE:

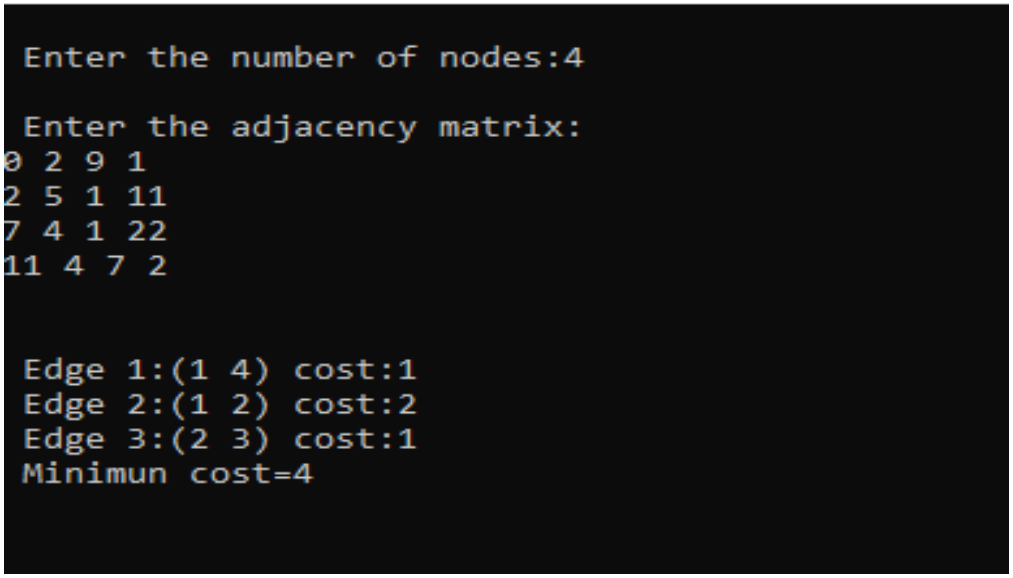
1. Create: Open Dev C++, write a program after that save the program with .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]= {0}
,min,mincost=0,cost[10][10];
void main()
 {
        printf("\n Enter the number of nodes:");
        scanf("%d",&n);
        printf("\n Enter the adjacency matrix:\n");
        for (i=1;i<=n;i++)
          for (j=1;j<=n;j++) {
                scanf("%d",&cost[i][j]);
                if(cost[i][j]==0)
                   cost[i][j]=999;
        }
        visited[1]=1;
        printf("\n");
        while(ne<n) {
                for (i=1,min=999;i<=n;i++)
                  for (j=1;j<=n;j++)
                   if(cost[i][j]<min)
                    if(visited[i]!=0) {
                          min=cost[i][j];
                          a=u=i;
```

```
                b=v=j;
            }
            if(visited[u]==0 || visited[v]==0) {
                printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
                mincost+=min;
                visited[b]=1;
            }
            cost[a][b]=cost[b][a]=999;
        }
        printf("\n Minimun cost=%d",mincost);
        getch();
}
```

OUTPUT:



WEEK -14

Objective:

To obtain the topological ordering of vertices in a given digraph. Compute the transitive closure of a given directed graph using Warshall's algorithm.

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.

2. Compile: Alt + F9

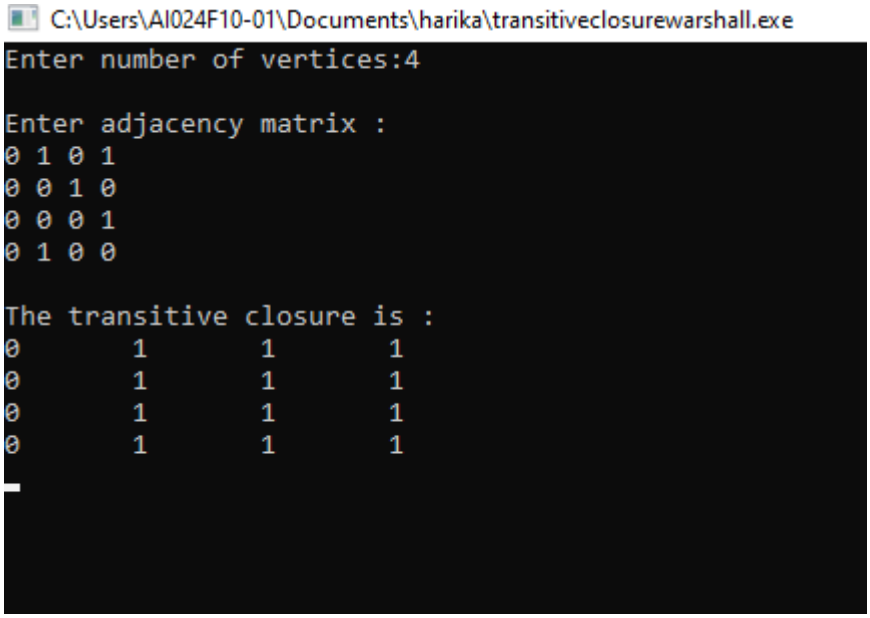
3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void warshall(int a[10][10], int n)
{
int i, j, k;
 for(k=0;k<n;k++)
 for(i=0;i<n;i++)
 if(a[i][k]==1)
  for(j=0;j<n;j++)
 a[i][j] = a[i][j] || a[k][j];
 }
 void main()
{
 int n, i, j, a[10][10];
 printf("Enter number of vertices:");
scanf("%d",&n);
printf("\nEnter adjacency matrix :\n");
 for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 scanf("%d",&a[i][j]);
warshall(a,n);
 printf("\nThe transitive closure is :\n");
for(i=0;i<n;i++)
{
```

```
 for(j=0;j<n;j++)
printf("%d\t",a[i][j]);
printf("\n");
}
getch();
 }
```

OUTPUT:



WEEK -15

Objective:

To Find a subset of a given set S = {s1,s2, ,sn} of n positive integers whose .....sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.

PROCEDURE:

1. Create: Open Dev C++, write a program after that save the program with .c extension.

2. Compile: Alt + F9

3. Execute: Ctrl + F10

SOURCE CODE:

```
#include<stdio.h>
int s[10],d,n,set[10],count=0;
```

```
void display(int);
int flag = 0;

void main()
{
        int subset(int,int);
        int i;
        printf("Enter the Number of elements in the set\n");
        scanf("%d",&n);
        printf("enter the set values\n");
        for(i=0;i<n;++i)
        scanf("%d",&s[i]);
        printf("\nEnter the sum\n");
        scanf("%d",&d);
        printf(" The Program Output is:\n");
        subset(0,0);
        if(flag == 0)
        printf(" There is no solution \n");
        getch();
}
int subset(int sum,int i)
{
        if(sum == d)
        {
                flag = 1;
                display(count);
                return;
        }
        if(sum>d || i>=n) return;
        else
        {
                set[count]=s[i];
                count++;
                subset(sum+s[i],i+1);
```

```
                count--;
                subset(sum,i+1);
        }
}
void display(int count)
{
        int i;
        printf("{ ");
        for(i=0;i<count;i++)
        printf("%d ",set[i]);
        printf("}\n");
}
```

OUTPUT: