

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institution Affiliated to VTU, Belagavi)

SHAVIGE MALLESHWARA HILLS, K.S.LAYOUT, BANGALORE-560078

Department of Computer Science and Engineering



2023-2024

IV Semester

Design and Analysis of Algorithms

Laboratory Manual

Compiled by

**Prof. A M Prasad
Prof. Aruna S**

DAYANANDA SAGAR COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision and Mission of the Department

Vision Statement

To provide a vibrant learning environment in Computer Science and Engineering with focus on industry needs and research, for the students to be successful global professionals contributing to the society.

Mission Statement

- To adopt a contemporary teaching learning process with emphasis on hands on and collaborative learning
- To facilitate skill development through additional training and encourage student forums for enhanced learning.
- To collaborate with Industry partners and Professional Societies and make the students industry ready.
- To encourage innovation through multidisciplinary research and development activities
- To inculcate human values and ethics in students and groom them to be responsible citizens.

Part – A

Bridge Course Exercises.

These are the problems are given to students in first two-week laboratory sessions. These are the exercises which are the prerequisite and align with the regular subject syllabus. The complete and formal solutions are provided for these problems.

#	Problem statement
1.	Using Euclid's method, write and execute "C" program to find Greatest common divisor of two given integer numbers.
2.	Using Integer check method, write and execute "C" program to find Greatest common divisor of two given integer numbers.
3.	Write and execute "C" program to check given set of 'n' numbers in an array are unique or not.
4.	Write and execute "C" program to find biggest of given set of 'n' numbers in an array.
5.	Write and execute "C" program to find product of two matrices of size NxN
6.	Write and execute "C" program to find factorial of a given number using recursion.
7.	Write and execute "C" program to generate Fibonacci numbers up to limit 'n'.
8.	Write and execute "C" program to find solution to Tower of Hanoi problem.
9.	Write and execute a program to sort a given set of elements using the Bubble sort method and determine the time required to sort the elements. Repeat the experiment for different values of n.
10.	Write and execute a program to sort a given set of elements using the Selection sort method and determine the time required to sort the elements. Repeat the experiment for different values of n.
11.	Write and execute a program to search given key element from given 'n' elements using the Binary search method.
12.	Using the Brute force string matching strategy, write and execute a program to search given pattern string in given text.

Part – B

These are the problems from regular syllabus. Students are required to

1. Understand problem.
2. Develop mathematical Solution.
3. Algorithm.
4. Program.

Support will be given only till algorithm development. The student required to convert algorithmic solution to the programmable solution. These problems are considered for laboratory semester end examination.

#	Problem statement	Unit
1	Write and execute a program to sort a given set of elements using the Quick sort method and determine the time required to sort the elements. 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 should be read from a file/can be generated using the random number generator	II
2	Write and execute a program to sort a given set of elements using the Merge sort method and determine the time required to sort the elements. 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 should be read from a file/can be generated using the random number generator	II
3	a. Write and execute a program to print all the nodes reachable from a given starting node in a graph using DFS method.	II
	b. Write and execute a program to print all the nodes reachable from a given starting node in a graph using BFS method.	
4	a. Write and execute a program to arrange nodes in topological order using DFS method.	II
	b. Write and execute a program to arrange nodes in topological order using source removal technique.	
5	Write and execute a program to search for the given pattern string in given text string using Horspool String Matching algorithm.	III
6	Write and execute a program for matrix chain multiplication for the given the sequence of Matrices $\langle A_1, A_2, \dots, A_n \rangle$.	III
7	Write and execute a program to find a longest-common-subsequence of X and Y using dynamic programming for given two sequences $X = \langle x_1, x_2, \dots, x_m \rangle$ $Y = \langle y_1, y_2, \dots, y_n \rangle$	III
8	Write and execute a program to find Minimum Spanning Tree using Prim's method	IV
9	Write and execute a program to find Minimum Spanning Tree using Kruskal's method	IV
10	Write and execute a program to find shortest path to all other nodes in weighted graph using Dijkstra's strategy	IV
11	Write and execute a program to find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ 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	V
12	Write and execute a program to find solution to n- queens problem	V

Part – A
Bridge Course Exercises

Question 1: Using Euclid's method, write and execute "C" program to find Greatest common divisor of two given integer numbers.

GCD of Two Numbers

The GCD (Greatest Common Divisor) of two numbers is the largest positive integer that divides both of them without leaving any remainder. It is the highest number that can evenly divide both numbers.

For example: GCD of (20, 30) = 10 (*10 is the largest number which divides 20 and 30 with remainder as 0*).

The GCD is used for a variety of applications in number theory, particularly in modular arithmetic and thus encryption algorithms such as RSA. It is also used for simpler applications, such as simplifying fractions.

Euclid Method to find GCD of two Numbers

In mathematics, the Euclidean algorithm is an efficient method for computing the greatest common divisor of two integers. It is named after the ancient Greek mathematician Euclid, who first described it in his Elements.

Mathematical Solution

The Euclidean Algorithm for finding GCD (A, B) is as follows:

- If $A = 0$ then $\text{GCD}(A, B) = B$, since the $\text{GCD}(0, B) = B$, and we can stop.
- If $B = 0$ then $\text{GCD}(A, B) = A$, since the $\text{GCD}(A, 0) = A$, and we can stop.

For example: Find the GCD of 78 and 36.

$A=78, B=36$

- $A \neq 0, B \neq 0$
- Use long division to find that $78/36 = 2$ with a remainder of 6.
- $78 = 36 * 2 + 6$
- Find $\text{GCD}(36,6)$, since $\text{GCD}(78,36) = \text{GCD}(36,6)$

$A=36, B=6$

- $A \neq 0, B \neq 0$
- Use long division to find that $36/6 = 6$ with a remainder of 0.
- $36 = 6 * 6 + 0$
- Find $\text{GCD}(6,0)$, since $\text{GCD}(36,6) = \text{GCD}(6,0)$

$A=6, B=0$

- $A \neq 0$
- $B = 0, \text{GCD}(6,0) = 6$

The GCD of 78 and 36 is computed as:

$\text{GCD}(78,36) = \text{GCD}(36,6) = \text{GCD}(6,0) = 6$

Ans = $\text{GCD}(270,192) = 6$

Euclid Algorithm

Euclid's algorithm for computing GCD (m, n)

Step 1 If $n = 0$, return the value of m as the answer and stop; otherwise, proceed to Step 2.

Step 2 Divide m by n and assign the value of the remainder to R.

Step 3 Assign the value of n to m and the value of r to n. Go to Step 1.

ALGORITHM Euclid (m, n)

```
//Computes GCD (m, n) by Euclid's algorithm
//Input: Two nonnegative, not-both-zero integers m and n
//Output: Greatest common divisor of m and n
```

```
while n!= 0 do
r ← m mod n
m ← n
n ← r
return m
```

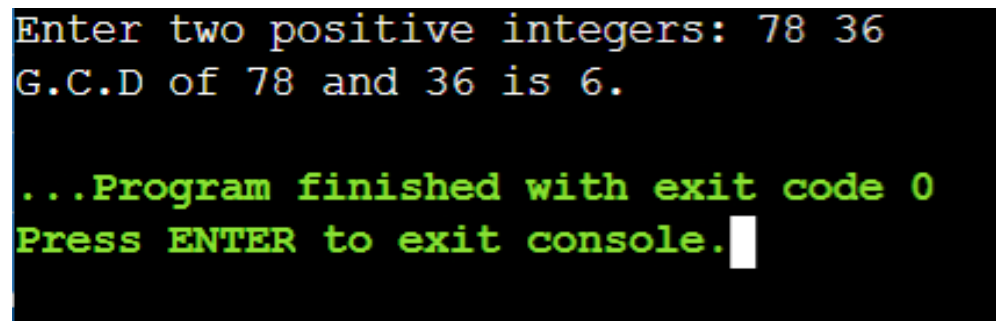
Program for Euclid Algorithm

```
#include <stdio.h>

//Function to find the GCD of 2 numbers
int GCD (int n1, int n2) {
if (n2!= 0)
return GCD (n2, n1 % n2);
else
return n1;
}

// main function
int main () {
int n1, n2;
printf("Enter two positive integers: ");
scanf("%d %d", &n1, &n2);
printf("G.C.D of %d and %d is %d.", n1, n2, GCD (n1, n2));
return 0;
}
```

Output:



```
Enter two positive integers: 78 36
G.C.D of 78 and 36 is 6.

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 2: Using Successive Integer Check method, write and execute “C” program to find Greatest common divisor of two given integer numbers.

Mathematical Solution:

The Successive Integer Check Algorithm for finding GCD (A, B) is as follows:

For example: Find the GCD of 78 and 36.

A=78, B=36

- $A \neq 0, B \neq 0$
- $A > B$, so minimum among these two is 36
- Check the modulus of 78 and 36, 36 and 36.
- $78 \% 36 = 6, 36 \% 36 = 0$.
- Since $36 \bmod 36$ is 0, repeat the process with 78 by decrementing 36 by 1.
- Process goes on till $78 \bmod 6$.
- Therefore, you get $\text{Ans} = \text{GCD}(78, 36) = 6$

Successive Integer Check Algorithm

Successive integer check algorithm for computing GCD (m, n)

Step 1: Assign the value of $\min(m, n)$ to t.

Step 2: Divide m by t. If the remainder of this division is 0, go to Step 3;

Otherwise, go to Step 4.

Step 3: Divide n by t. If the remainder of this division is 0, return the value of t as the answer and stop; otherwise, proceed to Step 4.

Step 4: Decrease the value of t by 1. Go to Step 2.

ALGORITHM IntCheck (m, n)

//Computes GCD (m, n) by Successive Integer Check Method

//Input: Two non-negative, not-both-zero integers m and n

//Output: Greatest common divisor of m and n

Begin

t = $\min(m, n)$

while (m mod t \neq 0 OR n mod t \neq 0)

t = t - 1

return t

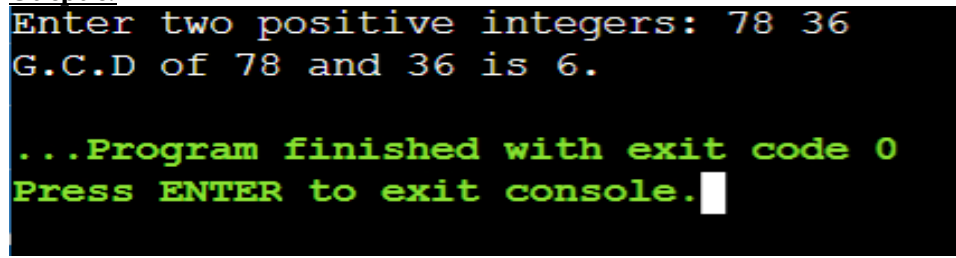
End

Program for Euclid Algorithm

```
#include <stdio.h>
// function to find the minimum of two numbers
int min(int n1, int n2)
{
    return ( n1<n2 ) ? n1: n2;
}
// function to find the GCD of two numbers
int GCD (int n1, int n2)
{
    t= min(n1,n2);
    while(n1 % t != 0 || n2 % t != 0)
    {
        t = t - 1;
    }
    return t;
}

// main function
int main ()
{
    int n1, n2, t=0;
    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("G.C.D of %d and %d is %d.", n1, n2, GCD (n1, n2));
    return 0;
}
```

Output:

A screenshot of a terminal window with a black background. The text is displayed in a monospaced font. The first two lines are in white: "Enter two positive integers: 78 36" and "G.C.D of 78 and 36 is 6.". The next two lines are in green: "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a white cursor block.

```
Enter two positive integers: 78 36
G.C.D of 78 and 36 is 6.

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 3: Write and execute “C” program to check given set of ‘n’ numbers in an array are unique or not.

Problem statement

Given an array of size n. Find the number of unique(non-repeating) elements in the array.

Logical Solution

Take the array size and elements as input from the user.

- Initialize an empty array called distinct.
- For each element in the input array:
 1. Check if the element is already in the unique array.
 2. If it is not, add it to the unique array.
- Print the unique array.

Algorithm to find UNIQUENESS(n)

The Algorithm to check the elements in the array are distinct.

The algorithm is given,

Algorithm Unique Elements (A[1....n])

 //Input: An array A[1....n]

 //Output: Returns “true” if all are distinct and “false” otherwise

for i ← 1 to n-1 do

begin

for j ← i+1 to n do

begin

if (a[i] = a[j]) return false

End

End

Return true

Program

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

```
//function to check uniqueness
```

```
int unique(int a[],int n)
{
    for(int i=0;i<n-1;i++)
        for(int j=i+1;j<n;j++)
        {
            if(a[i]==a[j])
                return 0;
        }
    return 1;
}
```

// main program

```
int main( )
{
    int n,a[100];
    printf("enter n elements of array\n");
    scanf("%d",&n);
    printf("enter the n elements\n");
    for(int i=0;i<n;i++)
        scanf("%d",&a[i]);
    if(unique(a,n))
        printf("elements of array are unique");
    else
        printf("elements of array are not unique");
}
```

Output

Run 1:

Enter n elements of array: 5
Enter the n elements: 3 5 7 9 11
Elements of array are unique

Run 2:

Enter n elements of array: 5
Enter the n elements: 1 5 5 3 8
Elements of array are not unique

Question 4: Write and execute “C” program to find biggest of given set of ‘n’ numbers in an array.

Algorithm:

1. Start the program.
2. Declare an array of an n elements
3. Initialize the array with n numbers
4. Declare a variable max and initialize it with the first element of the array
5. Use a for loop to iterate through the array from the second element to the last element
6. Inside the for loop compare each element of the array with max variable.
7. If the element is greater than the max variable update the max variable with element
7. After the for loop the max variable will contain the biggest number in array. 9. 8. Print the max variable as the output
10. End the program

Mathematical Solutions:

1. start
2. declare an array of n elements
3. initialise the array with n numbers
4. declare a variable max and initialize it with the first element of the array
5. for i=2 to n
6. if array[i]>max
7. max=array[i]
8. end if
9. end for
10. print max as the output
11. end

Program

```
#include<stdio.h>

void main()
{
    int i,n,a[100],large;

    printf("Enter the number of elements:\n") ;
    scanf("%d",&n) ;

    printf("Enter the elements\n") ;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]) ;
    }

    large=a[0];
    for(i=1;i<n;i++)
    {
        if(large<a[i])
        {
            large=a[i];
        }
    }
    printf("Largest of %d elements in an array = %d",n,large);
}
```

Question 5: Write and execute “C” program to find product of two matrices of size NxN

```
#include<stdio.h>

void main()
{
    int a[100][100];
    int i,j,n;
    printf("enter the order of matrix");
    scanf("%d",&n);
    printf("enter elements ");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("matrix of n*n order is ");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%d",a[i][j]);
        }
    }
}
```

Question 6: Write and execute “C” program to find factorial of a given number using recursion.

Problem statement

The factorial of a non-negative integer n , denoted by $n!$ is the product of all positive integers less than or equal to n . It is calculated as:

$$n! = n * (n-1) * (n-2) * \dots * 2 * 1$$

Mathematical Solution

To calculate the factorial of a given number, using the following recursive formula:

$$n! = n * (n-1)!$$

with the base case:

$$0! = 1$$

Algorithm FACTORIAL(n)

Input: An integer n

Output: Factorial of n

if($n=0$) then

return 1

else

return $n * \text{factorial}(n-1)$

Program

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

```
// Function to calculate factorial
```

```
int factorial(int n)
```

```
{
```

```
// Base case: factorial of 0 is 1
```

```
if (n == 0)
```

```
return 1;
```

```
else
```

```
return n * factorial(n-1);
```

```
}
```

```
int main()
{
int n;
printf("Enter a non-negative integer: ");
scanf("%d", &n);
int result = factorial(n);
printf("Factorial of %d is %d\n", n, result);
return 0;
}
```

Output

Enter a non-negative integer: 5

Factorial of 5 is 120

Question 7: Write and execute “C” program to generate Fibonacci numbers up to limit ‘n’.

Problem statement

The Fibonacci series is a sequence of numbers in which each number is the sum of the two preceding ones. It starts with 0 and 1.

Mathematical Solution

To generate the Fibonacci series, using the following recursive formula:

$$\text{Fib}(0) = 0$$

$$\text{Fib}(1) = 1$$

$$\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \text{ for } n > 1$$

Algorithm to find FIBONACCI(n)

Input: An integer n

Output: Printing the Fibonacci Sequence up to n value

Begin

If($n==0$ OR $n==1$)

Return n

Else

Return $\text{fib}(n-1) + \text{fib}(n-2)$

Print nth Fibonacci number

End

Program

```
#include <stdio.h>

int Fib(int n) {
    // base case
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return Fib(n-1) + Fib(n-2);
}

int main() {
    int n;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    // Generate the Fibonacci series up to n
    printf("Fibonacci series up to %d:\n", n);
    for (inti = 0; i<= n; i++) {
        printf("%d ", Fib(i));
    }
    return 0;
}
```

Output

```
Enter a positive integer: 7
Fibonacci series up to 7:
0 1 1 2 3 5 8 13
```

Question 8: Write and execute “C” program to find solution to Tower of Hanoi problem.

Problem statement

The Tower of Hanoi is a mathematical puzzle that consists of three pegs and a number of disks of different sizes. The puzzle starts with the disks stacked in ascending order of size on one peg, with the smallest disk at the top. The objective is to move the entire stack to another peg, following these rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the top disk from one of the stacks and placing it on top of another stack or on an empty peg.
3. No disk may be placed on top of a smaller disk.

LOGICAL SOLUTION

To solve the Tower of Hanoi problem recursively, using the following steps:

1. If the number of disks is 1, simply move it from the source peg to the destination peg.
2. Otherwise, recursively move $n-1$ disks from the source peg to the auxiliary peg.
3. Move the n th disk from the source peg to the destination peg.
4. Recursively move the $n-1$ disks from the auxiliary peg to the destination peg.

Algorithm TOWER_OF_HANOI(n , source, auxiliary, destination)

Input: Number of disks (n), source peg, auxiliary peg, destination peg

Output: Sequence of moves to solve the Tower of Hanoi puzzle

1. IF n is equal to 1 THEN

 Print "Move disk from" source "to" destination

ELSE

 Recursively call TOWER_OF_HANOI with ($n-1$, source, destination, auxiliary)

 Print "Move disk from" source "to" destination

 Recursively call TOWER_OF_HANOI with ($n-1$, auxiliary, source, destination)

END

Program

```
#include <stdio.h>

void towerofhanoi(int n, char source, char auxiliary, char destination)
{
    if (n == 1) {
        printf("Move disk from %c to %c\n", source, destination);
    } else {
        towerofhanoi(n-1, source, destination, auxiliary);
        printf("Move disk from %c to %c\n", source, destination);
        towerofhanoi(n-1, auxiliary, source, destination);
    }
}

int main() {
    int n;

    printf("Enter the number of disks: ");
    scanf("%d", &n);

    printf("Tower of Hanoi moves:\n");
    towerOfHanoi(n, 'A', 'B', 'C');

    return 0;
}
```

Output

```
Enter the number of disks: 3

Tower of Hanoi moves:
Move disk from A to C
Move disk from A to B
Move disk from C to B
Move disk from A to C
Move disk from B to A
Move disk from B to C
Move disk from A to C
```

Program 9. Write and execute a program to sort a given set of elements using the Bubble sort method and determine the time required to sort the elements. Repeat the experiment for different values of n.

Mathematical approach

Bubble sort algorithm takes the array of numbers as input and compares the adjacent numbers and performs swapping of numbers which are not in order and prints the array of numbers in ascending order.

Algorithm

Algorithm Bubble Sort [This algorithm takes a list of unsorted numbers and arranges them in ascending order using Bubble Sort Method].

Step 1: [Initialize] Start

Step 2: [Input] Read n

Step 3: [Input Unsorted array] Read elements to array a[]

Step 4: Print elements of array a[]

Step 5: [Iterate array a[] in two loops. Outer loop gives number of pass. Inner loop does Swap task. In each pass, compare each pair of adjacent items. If former elements are greater than latter one, swap them.]

For each value i in array a[i] to n do

for each value j in array a[j] to n-1 **do** [Compare each pair of adjacent elements]

If (a[j]>a[j+1]) **then**

[Swap these elements using temp variable] temp a[j]

a[j] a[j+1]

a[j+1] temp

End if

End for End for

Step 6: Print array with sorted elements

Step 7: [Finished]

End.

Program

```
#include<stdio.h> int main()
{
int a[100], n, i , j, temp;
printf("Enter the number of elements:\n"); scanf("%d",&n);
printf("Enter the %d elements of array:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]); for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(a[j]>a[j+1]) /*Compares each pair of adjacent elements*/
{
temp=a[j]; /*Swap the elements using temp variable*/ a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("The sorted array is:\n"); /*Prints the sorted elements*/ for(i=0;i<n;i++)
printf("%d\n",a[i]); return 0;
}
```

Program 10. Write and execute a program to sort a given set of elements using the selection sort method and determine the time required to sort the elements. Repeat the experiment for different values of n.

Mathematical approach

Program

```
#include<stdio.h> int main()
{
int a[100], n, i , j, temp,minpos;

printf("Enter the number of elements:\n");
scanf("%d",&n);

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

for(i=0;i<n-1;i++)
{
minpos = i;
for(j=i+1;j<n-i-1;j++)
{
if(a[j]<a[minpos])
minpos = j;
}

temp=a[i]; /*Swap the elements using temp variable*/ a[j]=a[j+1];
a[i]=a[minpos];
a[minpos]=temp
}

printf("The sorted array is:\n"); /*Prints the sorted elements*/ for(i=0;i<n;i++)
printf("%d\n",a[i]); return 0;
}
```

Program 11: Write a C program that reads N integer numbers and search a key Element using Binary search technique.

Mathematical approach

Binary Search starts with the middle element of the sorted list.

1. [low, high] denotes the range in which element has to be present and [mid] denotes the middle element.
2. Initially low = 0, high = number_of_elements and mid = floor((low+high)/2).
3. If the middle element of the list is equal to the „key“ then we have found the position the specified value.
4. Else if the „key“ is greater than the middle element then the „key“ has to be present in the last half of the list.
5. Or if the „key“ is lesser than the middle element then the „input key“ has to be present in the first half of the list.
6. Hence, the search list gets reduced by half after each iteration.

Algorithm

Step 1: [input the number of items] read n

Step 2: [input N elements from keyboard] for i=0 to n read a[i] end for

Step 3: [input the item to be searched] read key

Step 4: [initialization] low=0 high=n-1 found=0

Step 5: [search using binary search method] while (low <=high && !found)

mid=(low+high)/2 if(key==a[mid]) found=1

else if (key<a[mid]) high=mid-1

else low=mid+1 end while

Step 6: if(found==1)

Print “key found at mid+1 position” Else

Print “key not found”

Step 7: Stop

Program

```

#include<stdio.h>
int main()
{
int a[100], n, i, key, low, mid=0, high, found=0; printf("Enter the number of elements:\n");
scanf("%d",&n);
printf("Enter %d elements in ascending order:\n",n); for(i=0;i<n;i++)
scanf("%d",&a[i]); //read n array elements in ascending order printf("Enter an element to
search\n");
scanf("%d",&key); //read key element to search low=0;
high=n-1; //initialize range of elements
while(low<=high && !found)
{
mid= (low+high)/2; //compute mid element position if(key==a[mid])
found=1; //key found at mid position
else if (key<a[mid]) // If the middle element is greater than key then key //has to be present in the
range [low,mid-1]
high=mid-1;
else //If the middle element (mid) is less than key then key //has to present in range [mid+1 ,
high],
low=mid+1;
}
if(found==1)
printf("key found at position %d\n",mid+1); else
printf("key not found\n"); return 0;
}

```

Program 12 Using the Brute force string matching strategy, write and execute a program to search given pattern string in given text. [This program using C++]

```
#include<iostream>
#include<cstdlib>
using namespace std;
int main()
{
int i,j,temp;
char str[100]="This is a pattern matching"; char substr[20]="pattern";
for(i=0;str[i]!='';i++)
{
j=0;
if(str[i]==substr[j])
{
temp=i+1;
while(str[i]==substr[j])
{
i++; j++;
}
if(substr[j]=="")
{
cout<<"The substring is present in given string at position "<<temp<<"n";
exit(0);
} else
{
i=temp;
temp=0;
}
}
}
if(temp==0)
cout<<"The substring is not present in given stringn";
return 0;
}
```