Exp. No :3.1(a)

Date :

BUBBLE SORTING

AIM:

To write a program to demonstrate how Bubble Sort can be implemented.

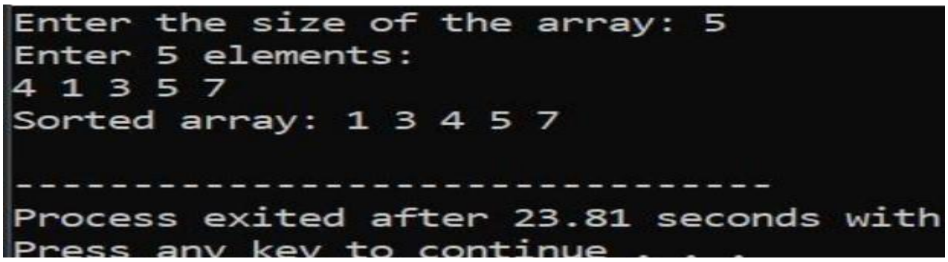
PSEUDOCODE:

```
 BEGIN
      Input n (size of the array)
          Input array of n elements
             for i from 0 to n - 1
              for j from 0 to n - i - 1
                   if array[j] > array[j + 1]
          swap array[j] and array[j + 1]
      print sorted array
 END
```

SOURCE CODE:

```
#include <stdio.h>
int main() {
int n;
  printf("Enter the size of the array: ");
  scanf("%d", &n);
int arr[n];
  printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
 for (int i = 0; i < n - 1; i++) {
  for (int j = 0; j < n - i - 1; j++) {
  if (arr[j] > arr[j + 1]) {
int temp = arr[j];
 arr[j] = arr[j + 1];
 arr[j + 1] = temp;
 }
}}
printf("Sorted array: ");
for (int i = 0; i < n; i++) {
   printf("%d ", arr[i]);
}
 printf("\n");
return 0;
}
```

OUTPUT:



```
Enter the size of the array: 5
Enter 5 elements:
4 1 3 5 7
Sorted array: 1 3 4 5 7

--------------------------------
Process exited after 23.81 seconds with
Press any key to continue . . .
```

RESULT:

Thus the program to bubble sorting was implemented and executed Successfully and the output is verified.

Exp. No :3.1(b) Date :	**INSERTION SORTING**

AIM:

To write a program to demonstrate how insertion Sort can be implemented.

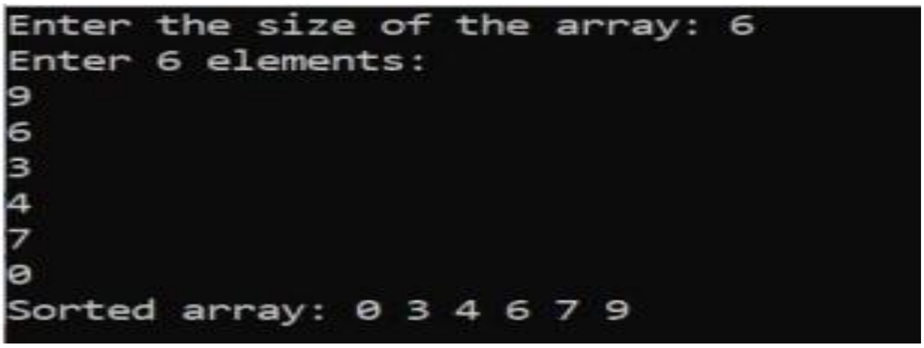
PSEUDOCODE:

```
BEGIN
     Input n (size of the array)
     Input array of n elements
  for i from 1 to n - 1
     key = array[i]
     j = i - 1
  while j >= 0 and array[j] > key
   array [j + 1] = array[j]
    j = j - 1
   array [j + 1] = key
   PRINT sorted array
END
```

SOURCE CODE:

```
#include <stdio.h>
 int main() {
  int n;
   printf("Enter the size of the array: ");
   scanf("%d", &n);
  int arr[n];
   printf("Enter %d elements:\n", n);
 for (int i = 0; i < n; i++) {
     scanf("%d", &arr[i]);
}for (int i = 1; i < n; i++) {
int key = arr[i];
  int j = i - 1;
while (j >= 0 && arr[j] > key) {
   arr[j + 1] = arr[j];
   j = j - 1;
}arr[j + 1] =key;
}
printf("Sorted array: ");
  for (int i = 0; i < n; i++) {
     printf("%d ", arr[i]);
    }
   printf("\n");
 return 0;
}
```

OUTPUT:



```
Enter the size of the array: 6
Enter 6 elements:
9
6
3
4
7
0
Sorted array: 0 3 4 6 7 9
```

RESULT:

Thus the program to insertion sorting was implemented and executed Successfully and the output is verified.

Exp. No :3.2(a) Date :	**MERGE SORT**

AIM:

To write a C-program to perform merge sort().
Data Structure used: Loop
Data Type: integer
Routine: Iterative

PSEUDOCODE:

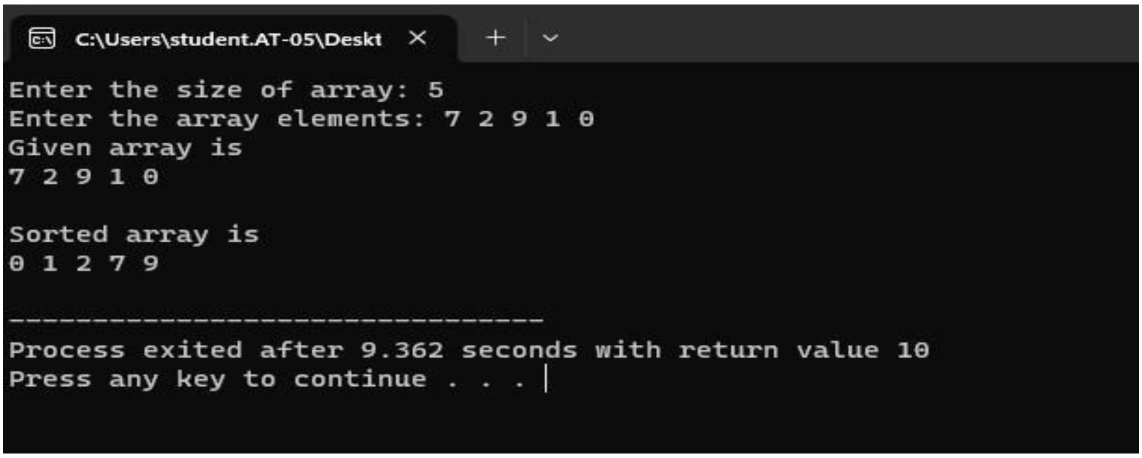
```
BEGIN
 while((l<=mid)&&(m<=high))
        {
                if(a[l]<=a[m])
                        temp[i++]=a[l++];
                else
                        temp[i++]=a[m++];
        }
        while(l<=mid)
                temp[i++]=a[l++]
        ; while(m<=high)
                temp[i++]=a[m++]
        ; for(k=low;k<=high;k++)
                a[k]=temp[k];
  END
```

SOURCE CODE:

```
        #include<stdlib.h> #include<stdio.h>
void merge(int a[], int low, int mid, int high)
{
int i,k,m,l,temp[50]; l=low;
i=low;
m=mid+1; while((l<=mid)&&(m<=high))
{
if(a[l]<=a[m])
temp[i++]=a[l++];
else
temp[i++]=a[m++];
}
while(l<=mid)
temp[i++]=a[l++]; while(m<=high)
temp[i++]=a[m++]; for(k=low;k<=high;k++)
a[k]=temp[k];
}
void partition(int arr[], int low, int high)
```

```
{
if (low < high)
{
int mid = (low+high)/2; partition(arr, low, mid); partition(arr, mid+1, high); merge(arr, low, mid, high);
}
}
void printArray(int A[], int size)
{
int i;
for (i=0; i < size; i++) printf("%d ", A[i]);
printf("\n");
}
void main()
{
int i,arr[20],n;
printf("Enter the size of array: "); scanf("%d",&n);
printf("Enter the array elements: "); for(i=0;i<n;i++)
scanf("%d",&arr[i]); partition(arr, 0, n-1); printf("\nSorted array is \n"); printArray(arr, n);
}
```

OUTPUT:



```
Enter the size of array: 5
Enter the array elements: 7 2 9 1 0
Given array is
7 2 9 1 0

Sorted array is
0 1 2 7 9

--------------------------------
Process exited after 9.362 seconds with return value 10
Press any key to continue . . .
```

RESULT:

Thus, the C program to sort the given array using merge sort( ) is successfully executed and the output is verified.

Exp. No :3.2(b) Date :	**QUICK SORT**

AIM:

To write a program for implementing the sort which is used in the following diagram.

PSEUDOCODE:

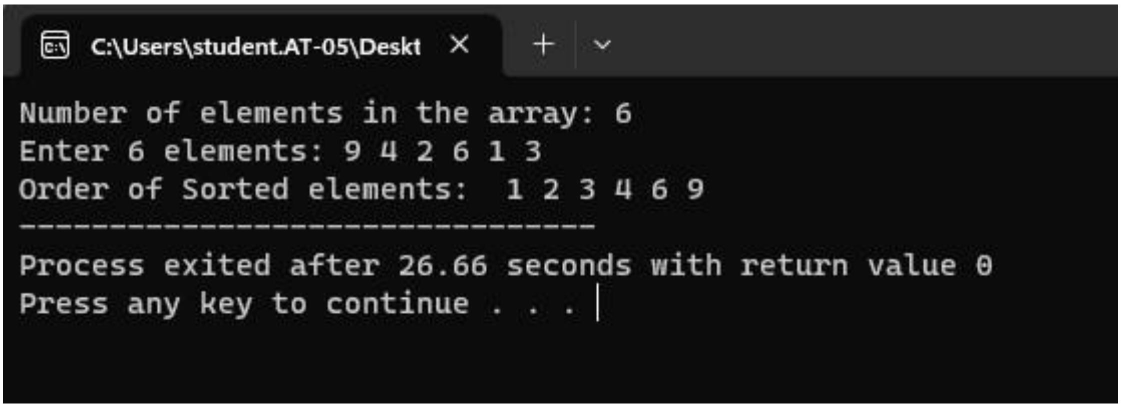
```
BEGIN

function partition Func (left, right,
pivot)left Pointer = left
right Pointer = right
- 1while True do
while A[++left Pointer]
<pivot do end while
while right Pointer > 0 && A[--right Pointer] >
pivotdo end while
if left Pointer >=
right Pointer break
else
swap leftPointer,rightPointer
end if
end while
swap
leftPointer,right
return leftPointer
end function
END
```

SOURCE CODE:

```
#include<stdlib.h>
#include<stdio.h>
void quicksort(int arr[25],int low,int high){
int i, j, pivot, temp, val;
 if(low<high){
pivot=low;
printf("Pivot =%d\n",pivot); i=low;
temp=arr[pivot];
arr[pivot]=arr[j];
arr[j]=temp;
quicksort(arr,low,j- 1);
quicksort(arr,j+1, high);
}
}
int main(){
int i, count, arr[25];
printf("Number of elements in tharray: ");
scanf("%d",&count);
printf("Enter %d elements: ", count);
for(i=0;i<count;i++)
     scanf("%d",&arr[i]);
     quicksort(arr,0,count-1);
printf("Order of Sorted elements: ");
for(i=0;i<count;i++)
        printf(" %d",arr[i]);
     return 0;
}
```

OUTPUT:



```
Number of elements in the array: 6
Enter 6 elements: 9 4 2 6 1 3
Order of Sorted elements:  1 2 3 4 6 9
--------------------------------
Process exited after 26.66 seconds with return value 0
Press any key to continue . . .
```

RESULT:

Thus, the C program to sorting using quick sort is successfully executed ant output is verified.

Exp. No :3.3(a) Date :	**LINEAR SEARCH**

AIM:

To write a C-program to perform linear search().
Data Structure used:Loop
Data Type: Integer
Routine: Iterative

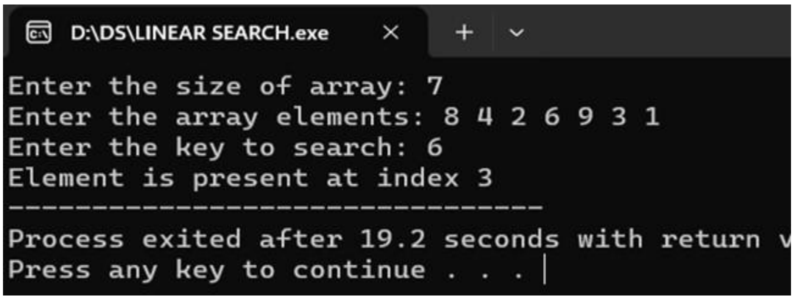
PSEUDOCODE:

```
BEGIN
            for(i=0;i<n;i++)
              if (arr[i] == x)
               return i;
             return -1;
END
```

SOURCE CODE:

```
#include <stdio.h>
 int linearSearch(int arr[], int n, int x)
{
 int i;
  for(i=0;i<n;i++)
    if (arr[i] == x)
     return i;
  return -1;
}
void main()
{
int arr[100],size,i,key;
   printf("Enter the size of array: ");
     scanf("%d",&size);
   printf("Enter the array elements: ");
     for(i=0;i<size;i++)
      scanf("%d",&arr[i]);
   printf("Enter the key to search: ");
      scanf("%d",&key);
   int result = linearSearch(arr, size, key);
(result == -1)? printf("Element is not present in array"):
   printf("Element is present at index %d", result);
}
```

OUTPUT:



```
Enter the size of array: 7
Enter the array elements: 8 4 2 6 9 3 1
Enter the key to search: 6
Element is present at index 3
--------------------------------
Process exited after 19.2 seconds with return v
Press any key to continue . . .
```

RESULT:

Thus the program to search the element in given array using linear search is executed Successfully and the output is verified.

Exp. No :3.3(b) Date :	**BINARY SEARCH**

AIM:

To write a C-program to perform binary search().

Data Structure used:Loop
Data Type: Integer
Routine: Iterative

PSEUDOCODE:

```
BEGIN
while (l <= r)
{
  int m = l + (r-l)/2;
 if (arr[m] == x)
     return m;
     if (arr[m] < x) l = m + 1;
  else
     r = m - 1;
}
  return -1;
END
```

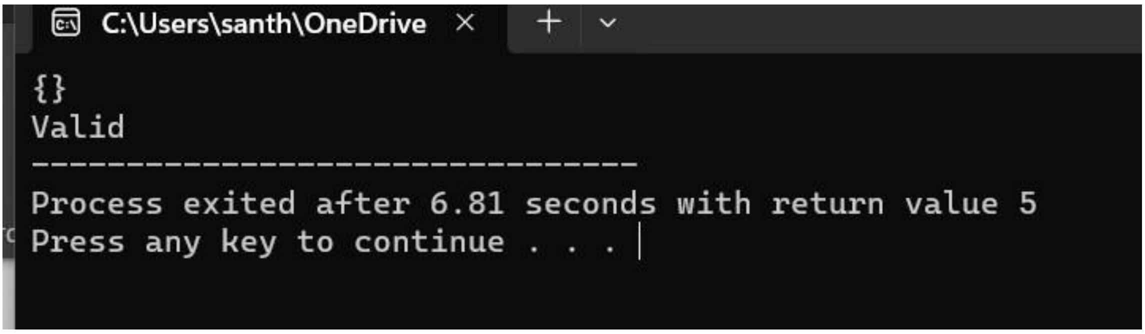
SOURCE CODE:

```
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
while (l <= r)
{
int m = l + (r-l)/2;
if (arr[m] == x)
  return m;
if (arr[m] < x) l = m + 1;
 else
   r = m - 1;
}
  return -1;
}
int main()
{
int arr[] = {2, 3, 4, 10, 40};
int n = sizeof(arr)/ sizeof(arr[0]);
int x = 10;
int result = binarySearch(arr, 0, n-1, x);
(result == -1)?
printf("Element is not present in array")
printf("Element is present at index %d", result);
return 0;
}
```

```
int check_parentheses(char exp[])
{
	int i;
	for(i=0;i<strlen(exp);i++)
	{
		if((exp[i]=='(')||(exp[i]=='{')||(exp[i]=='['))
			push(exp[i]);
		else if ((exp[i]==')')||(exp[i]=='}')||(exp[i]==']'))
		{

	if(((stack[top]=='(')&&(exp[i]==')'))||((stack[top]=='{')&&(exp[i]=='}'))||((stack[top]=='[')
&&(exp[i]==']')))
				pop();
			else //if(top==-1) or parenthesis din't match
				return 0;
		}
	}
	if(top==-1)
		return 1;
	else
		return 0;
}
void push(int element)
{
	stack[++top]=element;
}
void pop()
{
	top--;
}
```

OUTPUT:



```
{}
Valid
--------------------------------
Process exited after 6.81 seconds with return value 5
Press any key to continue . . .
```

Exp. No :3.4(a) Date :	**HASH TABLE USING LINEAR PROBING**

AIM:

To write a program to insert the given keys into the hash table using linear probing.

PSEUDOCODE:

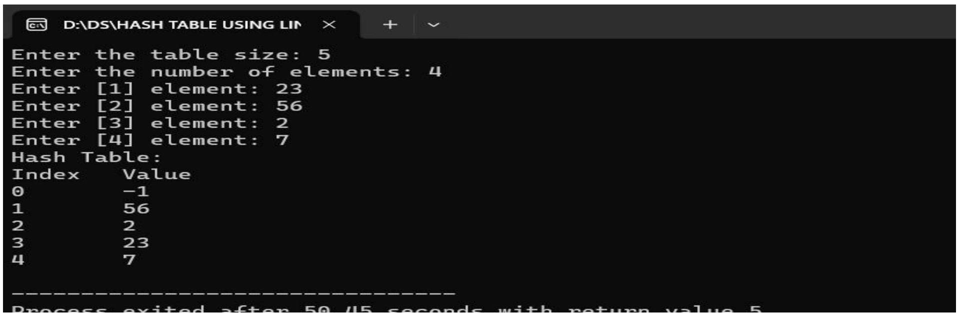
```
BEGIN
                    Get the element to insert in
        the hash table a=0
        while(1)
                hash_index=((element%table_size)+a)%table_size
                if(hash_table[hash_index]==-1)
                        hash_table[hash_index]=element
                        break
                else
                        Increment a
                end
        if End while
END
```

SOURCE CODE:

```
#include<stdio.h>
void LinearProbing(int arr[],int n,int table_size)
{
int i,element,j,HashKey;
  for(i=0;i<n;i++)
   {
     printf("Enter [%d] element: ",i+1);
     scanf("%d",&element);
     j=0;
while(1)
{
HashKey=((element % table_size) + j ) % table_size;
  if(arr[HashKey]==-1)
   {
    arr[HashKey]=element; break;
    }
    else j++;
  }
}
printf("Hash Table:\n");
   printf("Index Value\n");
   for(i=0;i<table_size;i++)
     printf("%d%d\n",i,arr[i]);
}
```

```
void main()
{
int table_size,i,n;
 int arr[10];
  printf("Enter the table size: ");
  scanf("%d",&table_size);
printf("Enter the number of elements: ");
 scanf("%d",&n);
 for(i=0;i<table_size;i++)
  arr[i]=-1;
 LinearProbing(arr,n,table_size);
}
```

OUTPUT:



```
Enter the table size: 5
Enter the number of elements: 4
Enter [1] element: 23
Enter [2] element: 56
Enter [3] element: 2
Enter [4] element: 7
Hash Table:
Index    Value
0        -1
1        56
2        2
3        23
4        7

--------------------------------
Process exited after 50.45 seconds with return value 5
```

RESULT:

Thus the program to insert keys into the hash table using linear probing is executed Successfully and the output is verified.

Exp. No :3.4(b) Date :	**HASH TABLE USING QUADRATIC PROBING**

AIM:

To write a program to insert the given keys into the hash table using quadratic probing.

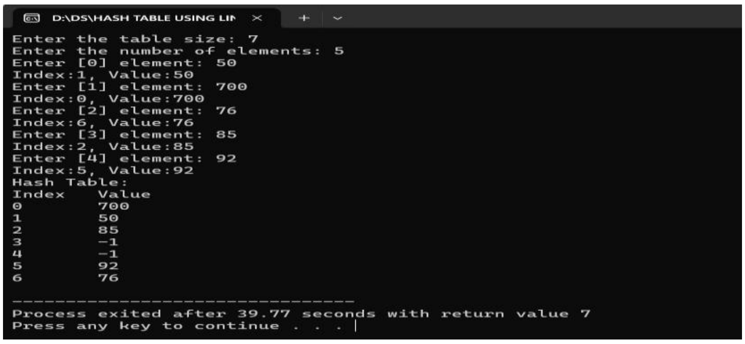
PSEUDOCODE:

```
BEGIN
        Get the element to insert in the hash table
        a=0
        while(1)
                hash_index=((element%table_size)+a*a)%table_size
                if(hash_table[hash_index]==-1)
                        hash_table[hash_index]=element
                        break
                else
                        Increment a
                end
        if End while
END
```

SOURCE CODE:

```
#include<stdio.h>
void QuadraticProbing(int arr[],int n,int table_size)
{
int i,element,j,HashKey;
  for(i=0;i<n;i++)
  {
    printf("Enter [%d] element: ",i);
    scanf("%d",&element);
    j=0;
   while(1)
    {
    HashKey=((element % table_size) + (j*j) ) % table_size;
   if(arr[HashKey]==-1)
    {
   arr[HashKey]=element; break;
   }
    else
      j++;
   }
      printf("Index:%d, Value:%d\n",HashKey,arr[HashKey]);
}
printf("Hash Table:\n");
printf("Index Value\n");
   for(i=0;i<table_size;i++)
   printf("%d%d\n",i,arr[i]);
}void  main(){
int table_size,i,n;
int arr[10];
printf("Enter the table size: ");
scanf("%d",&table_size);
printf("Enter the number of elements: ");
scanf("%d",&n);
for(i=0;i<table_size;i++)
arr[i]=-1;
 QuadraticProbing(arr,n,table_size);
}
```

OUTPUT:



```
Enter the table size: 7
Enter the number of elements: 5
Enter [0] element: 50
Index:1, Value:50
Enter [1] element: 700
Index:0, Value:700
Enter [2] element: 76
Index:6, Value:76
Enter [3] element: 85
Index:2, Value:85
Enter [4] element: 92
Index:5, Value:92
Hash Table:
Index   Value
0       700
1       50
2       85
3       -1
4       -1
5       92
6       76

--------------------------------
Process exited after 39.77 seconds with return value 7
Press any key to continue . . .
```

RESULT:

Thus the program to insert keys into the hash table using quadratic probing is executed Successfully and the output is verified.

Exp. No :3.4(c) Date :	**HASH TABLE USING DOUBLE PROBING**

AIM:

To write a program to insert the given keys into the hash table using double probing.

PSEUDOCODE:

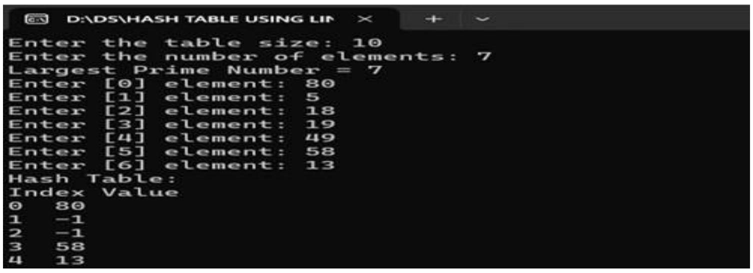
```
    BEGIN
        Get the element to insert in the hash table
        a=0
        while(1)
            HashKey=((element % table_size) + (j*(R-(element % R))) ) %
            table_size if(hash_table[hash_index]==-1)
                hash_table[hash_index]=element
                break
            else
                Increment a
            end
        if End while
END
```

SOURCE CODE:

```
#include<stdio.h>
void DoubleProbing(int arr[],int n,int table_size)
{
int i,element,j,HashKey,R,flag,t;
  for(R=table_size-1;R>1;R--)
   {
     flag=0;
  for(t=2;t<R;t++)
  {
    if(R%t ==0)
    {
      flag=1; break;
    }}
    if(flag==0)
     break;
    }
printf("Largest Prime Number = %d\n",R);
for(i=0;i<n;i++)
{
printf("Enter [%d] element: ",i);
scanf("%d",&element);
j=0;
```

```
 while(1){
    HashKey=((element % table_size) + (j*(R-(element % R))) ) %table_size;
if(arr[HashKey]==-1)
{
arr[HashKey]=element;
 break;
}else
  j++;}
 printf("Hash Table:\n"); printf("Index Value\n"); for(i=0;i<table_size;i++)
 printf("%d %d\n",i,arr[i]);}
 void main(){
   int table_size,i,n;
   int arr[10];
     printf("Enter the table size: ");
      scanf("%d",&table_size);
     printf("Enter the number of elements: ");
    scanf("%d",&n);
    for(i=0;i<table_size;i++)
     arr[i]=-1;
 DoubleProbing(arr,n,table_size);
}
```

OUTPUT:



```
Enter the table size: 10
Enter the number of elements: 7
Largest Prime Number = 7
Enter [0] element: 80
Enter [1] element: 5
Enter [2] element: 18
Enter [3] element: 19
Enter [4] element: 49
Enter [5] element: 58
Enter [6] element: 13
Hash Table:
Index Value
0   80
1   -1
2   -1
3   58
4   13
```

RESULT:

Thus the program of printing the conversion of postfix and evaluation of postfix expression is executed and the output is verified successfully.

.