Traversing of a static array:

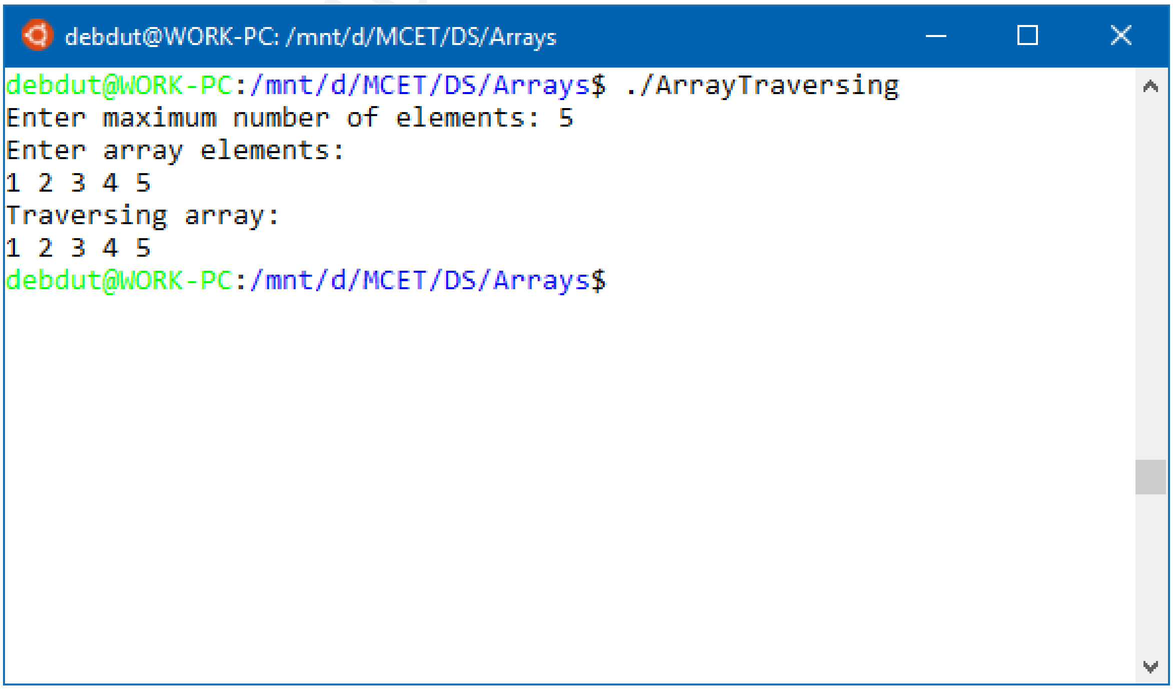
Algorithm:

```
Step1: Start

Step2: Declare a static array a[20].

Step3: Scan length 'n'

Step4: Run a for loop from 0 to n-1
           Scan each element
           Insert into array

Step5: Run a for loop from 0 to n-1
           Print each element

Step6: End
```

Source Code:

```c
/*Traversing Static Array
  Date: 18.07.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

void main()
  {
      int input_arr[20],n,i;
      system("clear");

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

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

      printf("Traversing array:\n");
      for(i=0;i<n;i++)
          printf("%d ",input_arr[i]);

      printf("\n");
  }
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$ ./ArrayTraversing
Enter maximum number of elements: 5
Enter array elements:
1 2 3 4 5
Traversing array:
1 2 3 4 5
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$
```

Insert an element to the beginning and end of an array:

Algorithm:

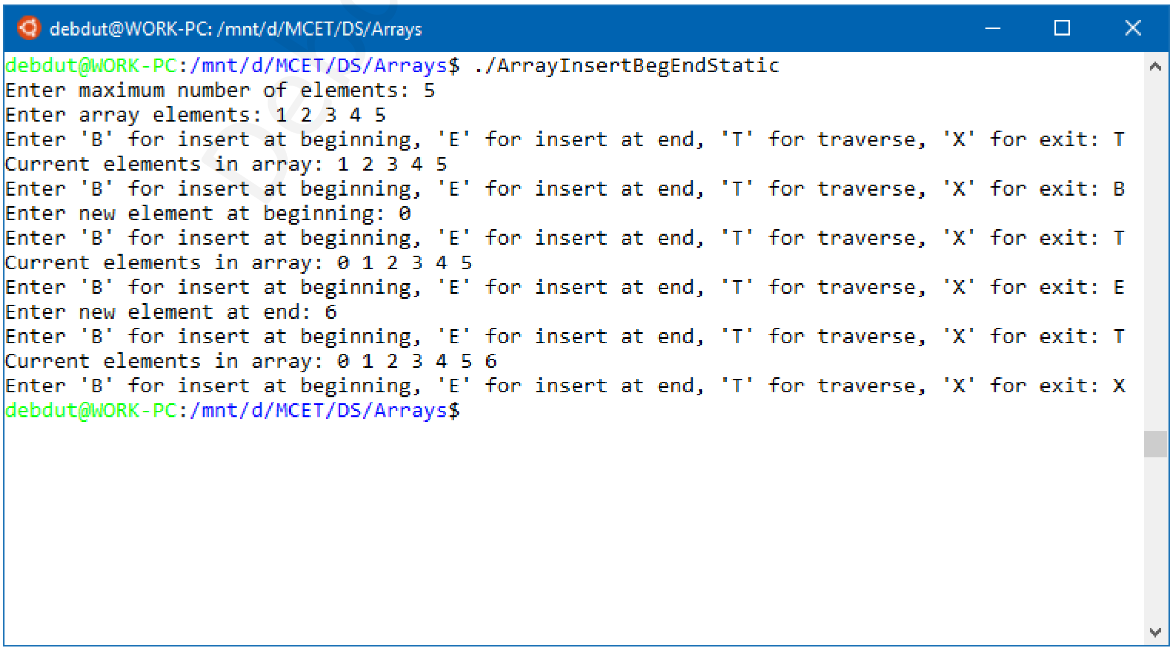
```
Step1:  Start
Step2:  Declare an array a[n]
        'n' number of elements
        'i' as increment variable
        'newElement' to store new element temporarily
Step3:  In main function declare choice as char choice variable
        Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
        Run an infinite while loop
            Scan 'choice'
            switch case choice variable
                case 'B': Call insertBeg()
                case 'E': Call insertEnd()
                case 'T': Call traverse()
                default: return
Step4:  Define function insertBeg()
            Scan new element into 'newElement'
            Run a for loop from n to 0
                set a[i]=a[i-1]
            set a[0]=newElement
            increment 'n' by 1
Step5:  Define function insertEnd()
            Scan new element into 'newElement'
            set a[n]=newElement
            increment 'n' by 1
Step6:  Define function traverse()
            Run a for loop from 0 to n-1
                print a[i]
Step7:  End
```

Source Code:

```c
/*Inserting Elements To Beginning and End of a Static Array
  Date: 26.07.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i,newElement;

void insertBeg();
void insertEnd();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'B': insertBeg();
                    break;

            case 'E': insertEnd();
                    break;

            case 'T': traverse();
                    break;

            default: return;
        }
    }
}



//P.T.O.
```

```c

void insertBeg()
{
    printf("Enter new element at beginning: ");
    scanf("%d",&newElement);

    for(i=n;i>=0;i--)
    {
        input_arr[i]=input_arr[i-1];
    }

    input_arr[0]=newElement;
    n++;
}

void insertEnd()
{
    printf("Enter new element at end: ");
    scanf("%d",&newElement);

    input_arr[n]=newElement;
    n++;
}

void traverse()
{
    printf("Current elements in array: ");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$ ./ArrayInsertBegEndStatic
Enter maximum number of elements: 5
Enter array elements: 1 2 3 4 5
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 3 4 5
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: B
Enter new element at beginning: 0
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 0 1 2 3 4 5
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: E
Enter new element at end: 6
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 0 1 2 3 4 5 6
Enter 'B' for insert at beginning, 'E' for insert at end, 'T' for traverse, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$
```

Insert an element to the given position of an array:

Algorithm:

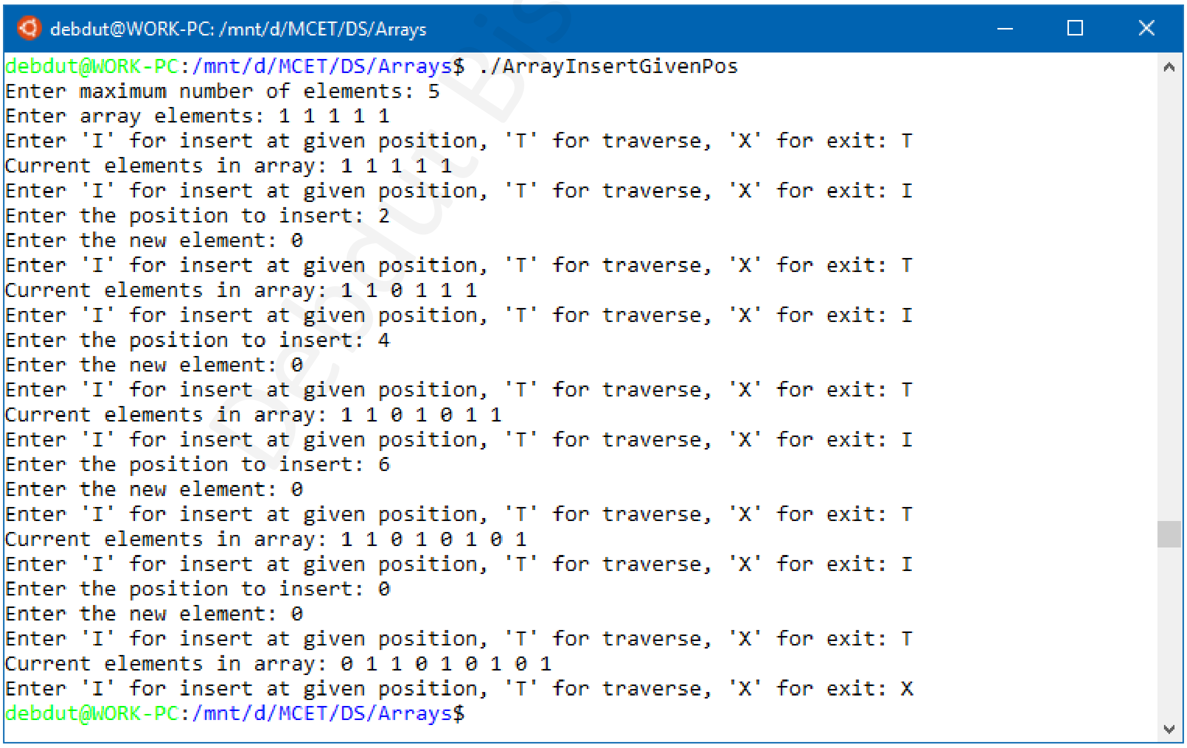
```
Step1:  Start
Step2:  Declare an array a[n]
        ‘n’ number of elements
        ‘i’ as increment variable
        ‘elementPos’ to store new element location temporarily
        ‘newElement’ to store new element temporarily
Step3:  In main function declare choice as char variable for choice
        Scan ‘n’
        Run a for loop from 0 to n-1
            scan ‘a[i]’
        Run an infinite while loop
            scan ‘choice’
            switch case ‘choice’ variable
                case ‘I’: call insertGivenPos()
                case ‘T’: call traverse()
                default: return
Step4:  Define function insertGivenPos()
            Scan element position into ‘elementPos’
            Scan new element into ‘newElement’
            Run a for loop from n to elementPos
                set a[i]=a[i-1]
            set a[elementPos]=newElement
            increment n by 1
Step5:  Define function traverse()
            Run a for loop from 0 to n-1
                print a[i]
Step6:  End
```

Source Code:

```
/*Inserting Elements To Given Position of a Static Array
  Date: 09.08.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i,elementPos,newElement;

void insertGivenPos();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'I': insertGivenPos();
                    break;

            case 'T': traverse();
                    break;

            default: return;
        }
    }
}








//P.T.O.
```

```c

void insertGivenPos()
{
    printf("Enter the position to insert: ");
    scanf("%d",&elementPos);
    printf("Enter the new element: ");
    scanf("%d",&newElement);

    for(i=n;i>=elementPos;i--)
    {
        input_arr[i]=input_arr[i-1];
    }

    input_arr[elementPos]=newElement;
    n++;
}

void traverse()
{
    printf("Current elements in array: ");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$ ./ArrayInsertGivenPos
Enter maximum number of elements: 5
Enter array elements: 1 1 1 1 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 1 1 1 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: I
Enter the position to insert: 2
Enter the new element: 0
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 1 0 1 1 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: I
Enter the position to insert: 4
Enter the new element: 0
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 1 0 1 0 1 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: I
Enter the position to insert: 6
Enter the new element: 0
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 1 0 1 0 1 0 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: I
Enter the position to insert: 0
Enter the new element: 0
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 0 1 1 0 1 0 1 0 1
Enter 'I' for insert at given position, 'T' for traverse, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$
```

Delete an element from the given position of an array:

Algorithm:

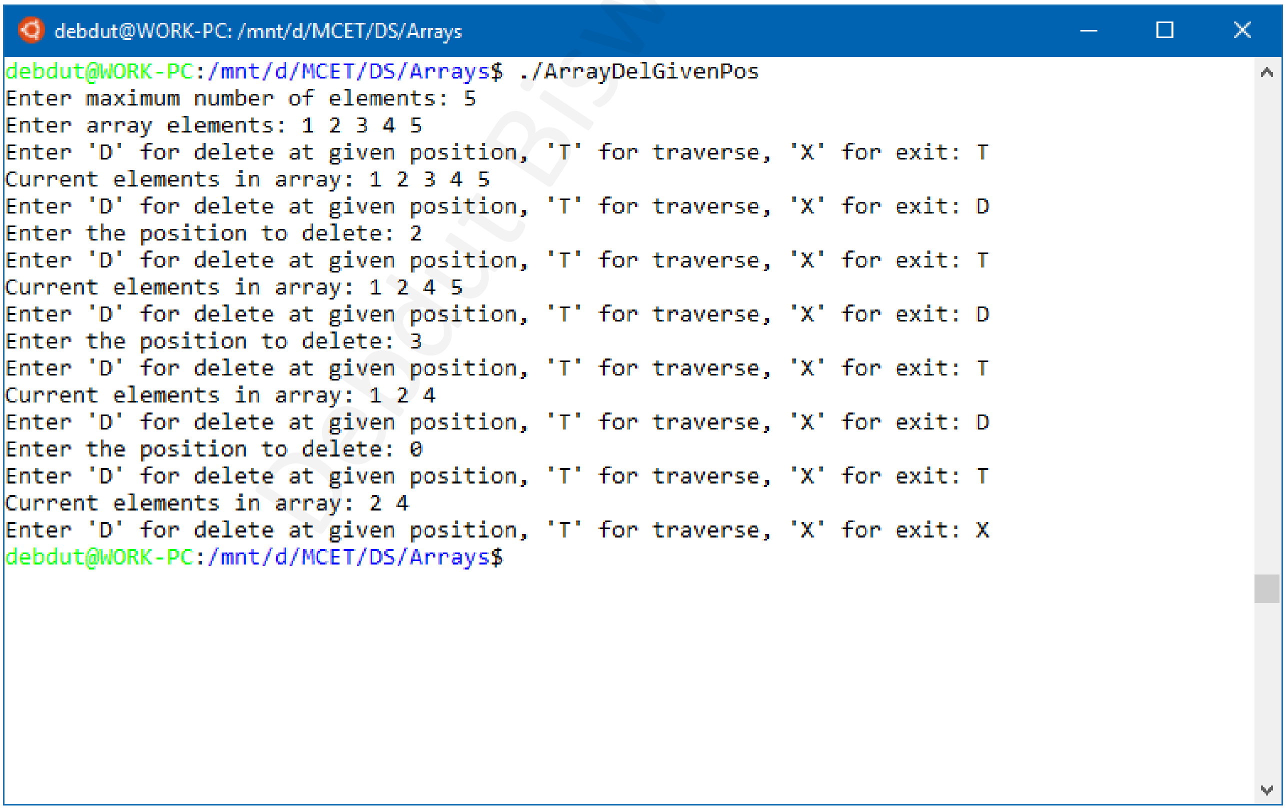
```
Step1:  Start
Step2:  Declare an array a[n]
        'n' number of elements
        'i' as increment variable
        'elementPos' to store deletion element location temporarily
Step3:  In main function declare choice as char variable for choice
        Scan 'n'
        Run a for loop from 0 to n-1
            scan 'a[i]'
        Run an infinite while loop
            scan 'choice'
            switch case 'choice' variable
                case 'D': call deleteGivenPos()
                case 'T': call traverse()
                default: return
Step4:  Define function deleteGivenPos()
            Scan element position into 'elementPos'
            Run a for loop from elementPos to n-1
                set a[i]=a[i+1]
            set a[n]=0
            decrement n by 1
Step5:  Define function traverse()
            Run a for loop from 0 to n-1
                print a[i]
Step6:  End
```

Source Code:

```c
/*Deletion Elements To Given Position of a Static Array
  Date: 09.08.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i,elementPos;

void deleteGivenPos();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'D': deleteGivenPos();
                    break;

            case 'T': traverse();
                    break;

            default: return;
        }
    }
}







//P.T.O.
```

```

void deleteGivenPos()
{
    printf("Enter the position to delete: ");
    scanf("%d",&elementPos);

    for(i=elementPos;i<n;i++)
    {
        input_arr[i]=input_arr[i+1];
    }

    input_arr[n]=0;
    n--;
}

void traverse()
{
    printf("Current elements in array: ");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$ ./ArrayDelGivenPos
Enter maximum number of elements: 5
Enter array elements: 1 2 3 4 5
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 3 4 5
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: D
Enter the position to delete: 2
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 4 5
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: D
Enter the position to delete: 3
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 4
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: D
Enter the position to delete: 0
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: T
Current elements in array: 2 4
Enter 'D' for delete at given position, 'T' for traverse, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$
```

Delete an element from the beginning and end of an array:

Algorithm:

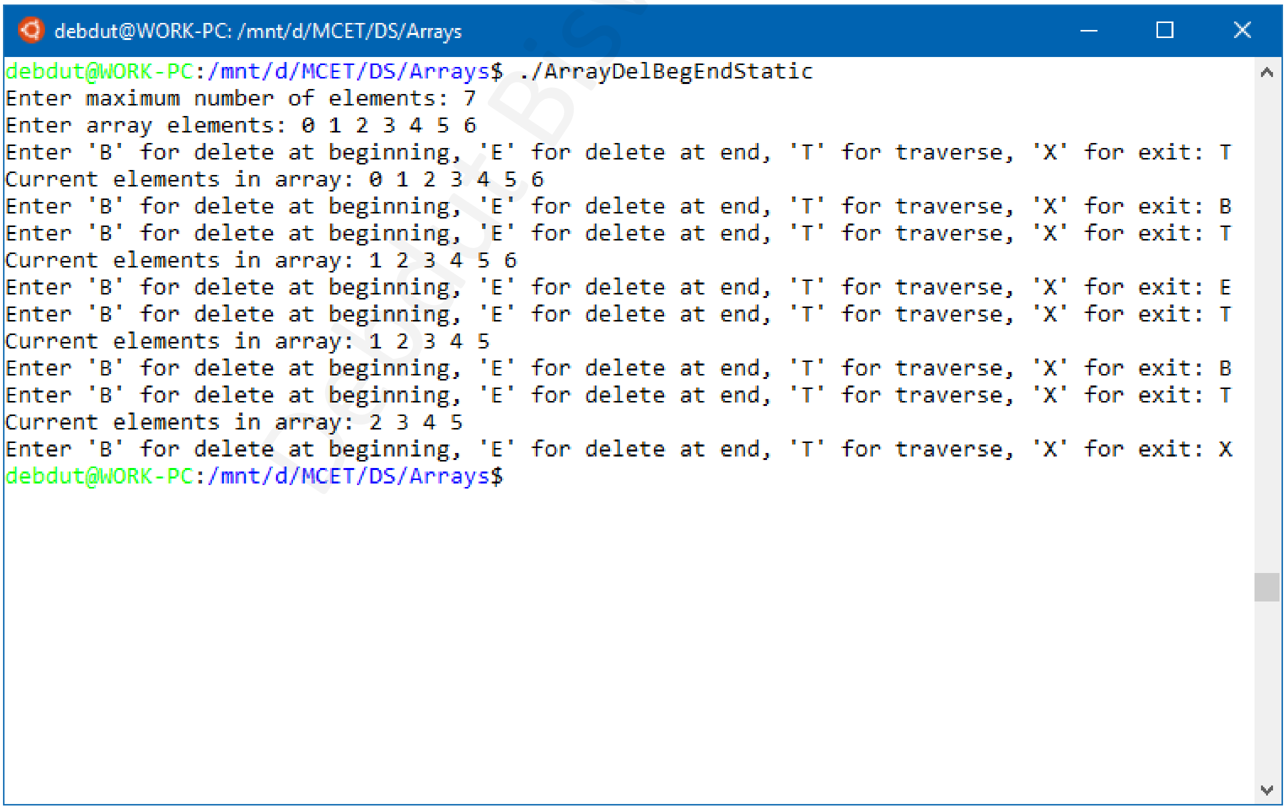
```
Step1:  Start
Step2:  Declare an array a[n]
        'n' number of elements
        'i' as increment variable
Step3:  In main function declare choice as char choice variable
        Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
        Run an infinite while loop
            Scan 'choice'
            switch case choice variable
                case 'B': Call deleteBeg()
                case 'E': Call deleteEnd()
                case 'T': Call traverse()
                default: return
Step4:  Define function deleteBeg()
            Run a for loop from 0 to n-1
                set a[i]=a[i+1]
            set a[n]=0
            decrement 'n' by 1
Step5:  Define function deleteEnd()
            set a[n]=0
            decrement 'n' by 1
Step6:  Define function traverse()
            Run a for loop from 0 to n-1
                print a[i]
Step7:  End
```

Source Code:

```c
/*Deleting Elements To Beginning and End of a Static Array
  Date: 09.08.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i;

void deleteBeg();
void deleteEnd();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'B': deleteBeg();
                    break;

            case 'E': deleteEnd();
                    break;

            case 'T': traverse();
                    break;

            default: return;
        }
    }
}



//P.T.O.
```

```c

void deleteBeg()
{
    for(i=0;i<n;i++)
    {
        input_arr[i]=input_arr[i+1];
    }

    input_arr[n]=0;
    n--;
}

void deleteEnd()
{
    input_arr[n]=0;
    n--;
}

void traverse()
{
    printf("Current elements in array: ");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$ ./ArrayDelBegEndStatic
Enter maximum number of elements: 7
Enter array elements: 0 1 2 3 4 5 6
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 0 1 2 3 4 5 6
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: B
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 3 4 5 6
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: E
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 1 2 3 4 5
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: B
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: T
Current elements in array: 2 3 4 5
Enter 'B' for delete at beginning, 'E' for delete at end, 'T' for traverse, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Arrays$
```

Linear Search:

Algorithm:

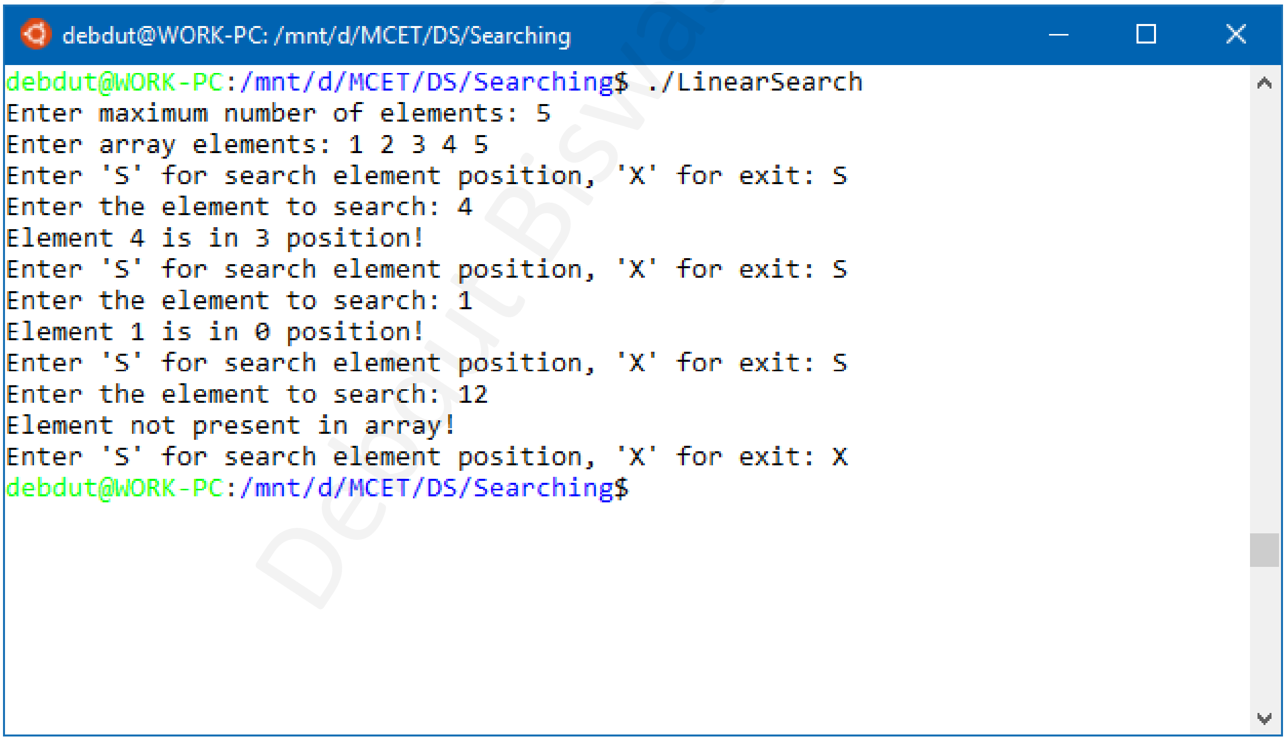
```
Step1:  Start
Step2:  Declare an array a[n]
        'n' number of elements
        'i' as increment variable
        'searchElement' to store element that will be searched temporarily
Step3:  In main function declare choice as char choice variable
        Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
        Run an infinite while loop
            Scan 'choice'
            switch case choice variable
                case 'S': linearSearch()
                default: return
Step4:  Define function linearSearch()
            Scan search element into 'searchElement'
            Run a for loop from 0 to n-1
                If a[i]=searchElement then
                    print 'i'
                    return
                print "element not present"
Step5:  End
```

Source Code:

```c
/*Linear Search
  Date: 30.08.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i,searchElement;

void linearSearch();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'S' for search element position, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'S': linearSearch();
                    break;

            default: return;
        }
    }
}











//P.T.O.
```

```c

void linearSearch()
{
    printf("Enter the element to search: ");
    scanf("%d",&searchElement);

    for(i=0;i<n;i++)
    {
        if(input_arr[i]==searchElement)
        {
            printf("Element %d is in %d position!\n",searchElement,i);
            return;
        }
    }
    printf("Element not present in array!\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Searching$ ./LinearSearch
Enter maximum number of elements: 5
Enter array elements: 1 2 3 4 5
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 4
Element 4 is in 3 position!
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 1
Element 1 is in 0 position!
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 12
Element not present in array!
Enter 'S' for search element position, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Searching$
```

Binary Search:

Algorithm:

```
Step1:  Start
Step2:  Declare an array a[n]
        ‘n’ number of elements
        ‘i’ as increment variable
        ‘searchElement’ to store element that will be searched temporarily
Step3:  In main function declare choice as char choice variable
        Scan ‘n’
        Run a for loop from 0 to n-1
            scan a[i]
        Run an infinite while loop
            Scan ‘choice’
            switch case choice variable
                case ‘S’: binarySearch()
                default: return
Step4:  Define function binarySearch()
            Declare and assign ‘left=0’ as left bound, ‘right=n-1’ as right
            Declare ‘mid’ as middle of bound
            ‘searchElement’ as element to be searched
            Scan search element into ‘searchElement’
            Run a while loop with condition ‘left<=right’
                set mid=(left+right)/2;
                if a[mid]<searchElement then
                    left=mid+1
                else if a[mid]>searchElement then
                    right=mid-1
                else
                    print ‘I’
                    return
                print “element not present”
Step5:  End
```

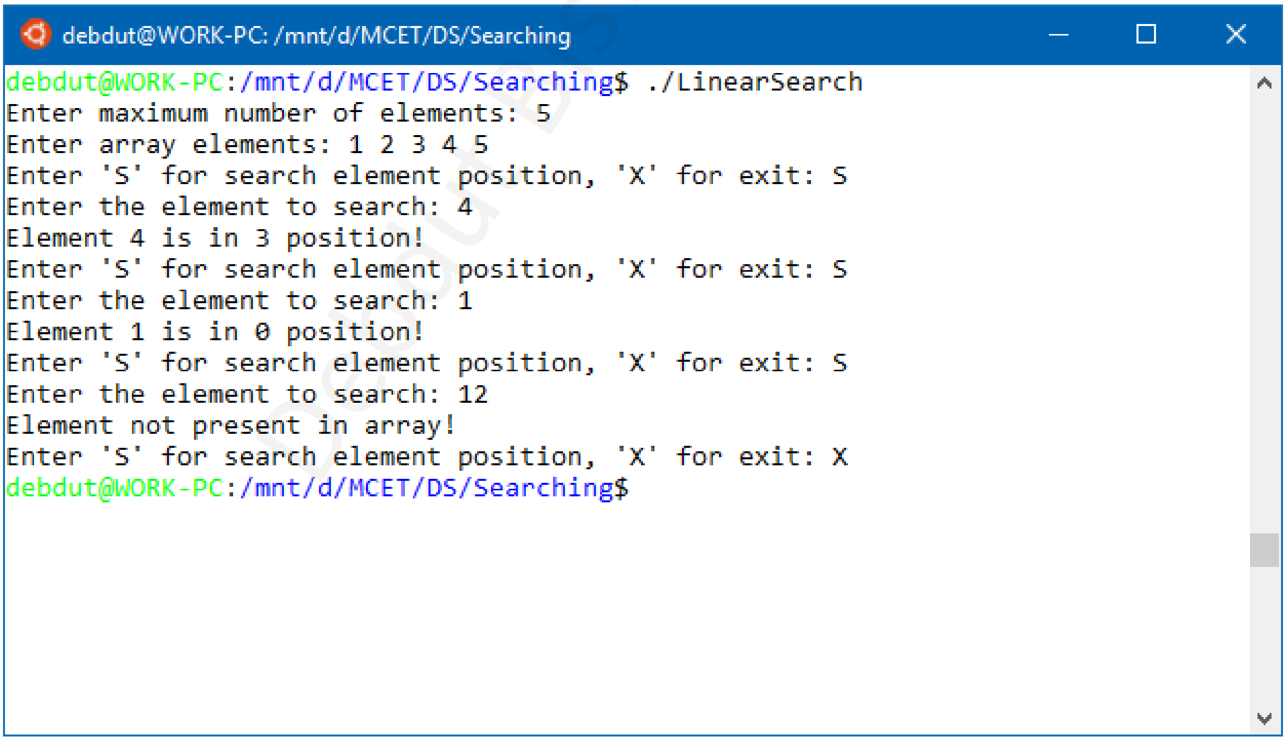
NOTE: BINARY SEARCH NEEDS A SORTED ARRAY TO BE SEARCHED!

Source Code:

```
/*Binary Search
  Date: 30.08.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int input_arr[20],n,i;

void binarySearch();
void traverse();

void main()
{
    char choice;
    printf("Enter maximum number of elements: ");
    scanf("%d",&n);

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

    while(1)
    {
        printf("Enter 'S' for search element position, 'X' for exit: ");
        scanf("%s",&choice);

        switch(choice)
        {
            case 'S': binarySearch();
                    break;

            default: return;
        }
    }
}










//P.T.O.
```

```c

void binarySearch()
{
    int left=0,right=n-1,mid,searchElement;

    printf("Enter the element to search: ");
    scanf("%d",&searchElement);

    while(left<=right)
    {
        mid=(left+right)/2;

        if(input_arr[mid]<searchElement)
            left=mid+1;
        else if(input_arr[mid]>searchElement)
            right=mid-1;
        else
        {
            printf("Element %d is in %d position!\n",searchElement,mid);
            return;
        }
    }
    printf("Element not present in array!\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Searching$ ./LinearSearch
Enter maximum number of elements: 5
Enter array elements: 1 2 3 4 5
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 4
Element 4 is in 3 position!
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 1
Element 1 is in 0 position!
Enter 'S' for search element position, 'X' for exit: S
Enter the element to search: 12
Element not present in array!
Enter 'S' for search element position, 'X' for exit: X
debdut@WORK-PC:/mnt/d/MCET/DS/Searching$
```

Selection Sort:

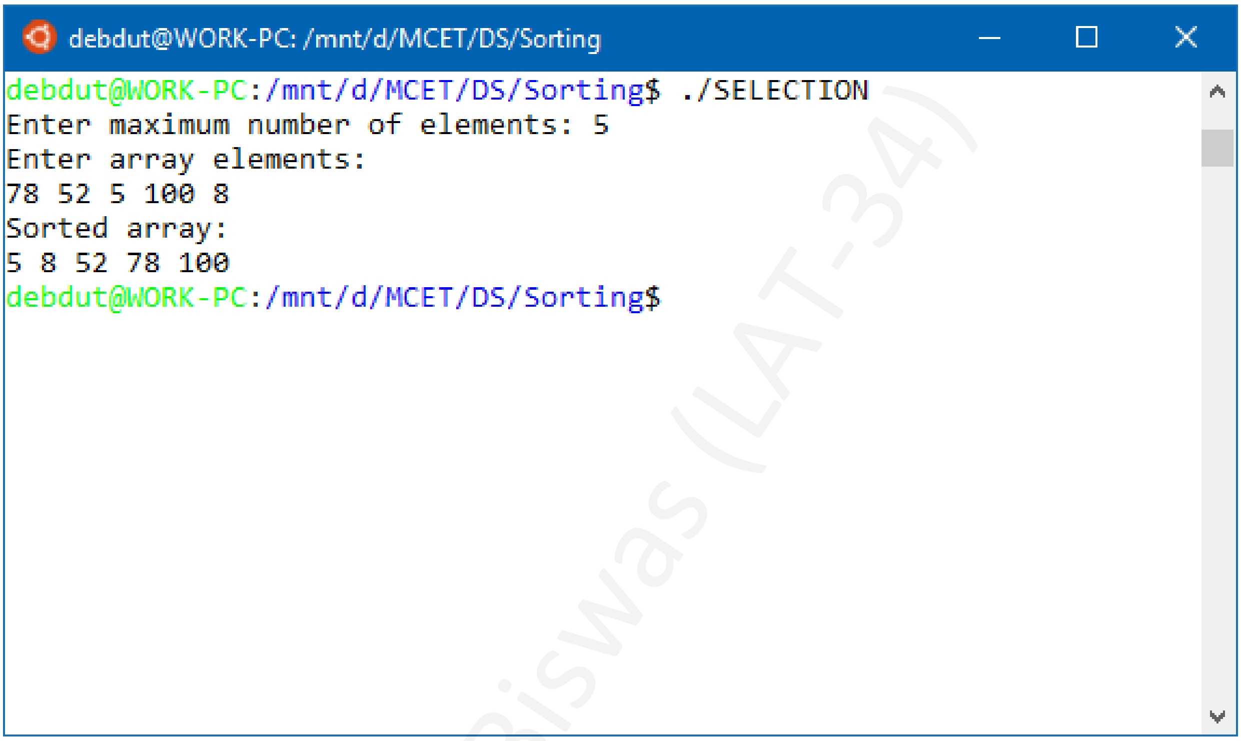
Algorithm:

```
Step1:  Start
Step2:  In main function declare an array a[n]
        'n' number of elements
        'i' and 'j' as loop variable
        'temp' to store temporary variable for swapping
        'smallest' to store smallest element of array
Step3:  Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
Step4:  Run a for loop from 0 to n-1, loop variable 'i'
            set 'smallest=i'
            run a for loop from i+1 to n-1, loop variable 'j'
                if 'a[j] < a[smallest]' then
                    smallest=j
            if 'smallest != i' then
                swap 'a[smallest]' with 'a[i]'
Step5:  Run a for loop from 0 to n-1
            print 'a[i]'
Step6:  End
```

Source Code:

```c
/*Selection Sort
  Date: 13.09.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

void main()
{
    int input_arr[20],n,i,j,temp,smallest;
    system("clear");

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

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

    for(i=0;i<n;i++)
    {
        smallest=i;

        for(j=i+1;j<n;j++)
        {
            if(input_arr[j]<input_arr[smallest])
                smallest=j;
        }

        if(smallest!=i)
        {
            temp=input_arr[smallest];
            input_arr[smallest]=input_arr[i];
            input_arr[i]=temp;
        }
    }

    printf("Sorted array:\n");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$ ./SELECTION
Enter maximum number of elements: 5
Enter array elements:
78 52 5 100 8
Sorted array:
5 8 52 78 100
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$
```

Bubble Sort:

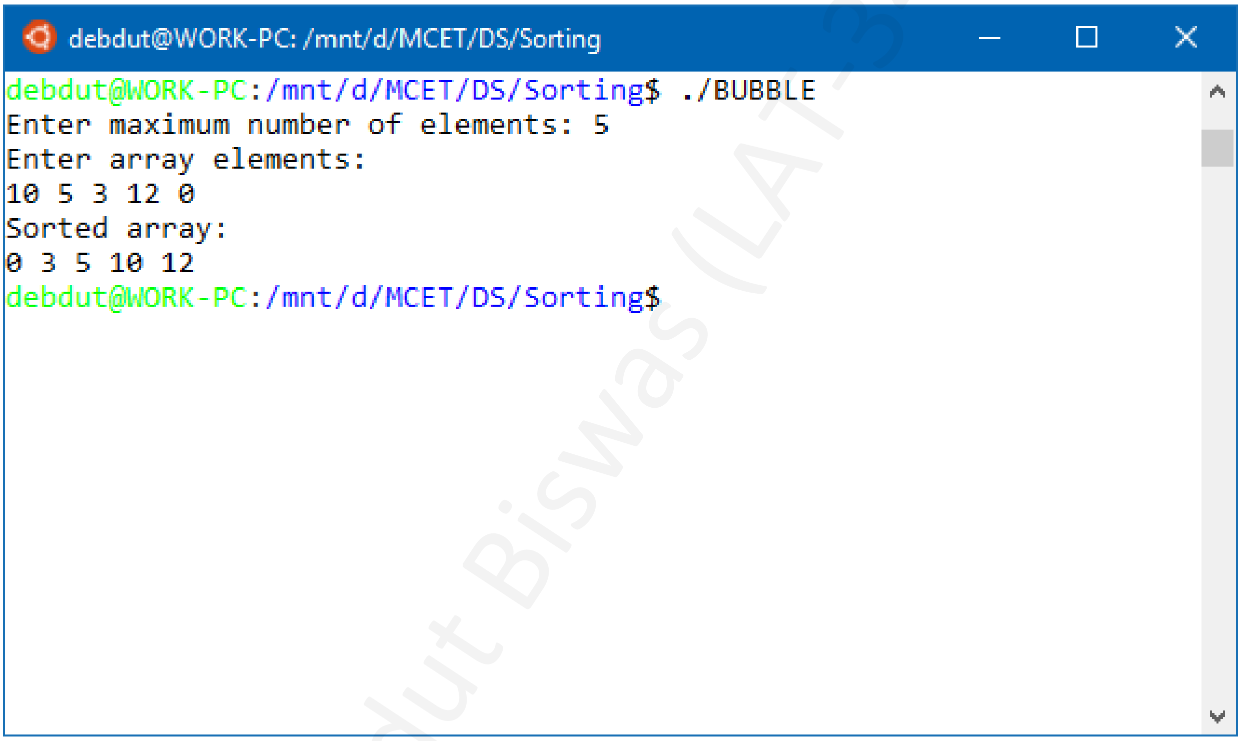
Algorithm:

```
Step1:  Start
Step2:  In main function declare an array a[n]
        'n' number of elements
        'i' and 'j' as loop variable
        'temp' to store temporary variable for swapping
Step3:  Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
Step4:  Run a for loop from 0 to n-1, loop variable 'i'
            run a for loop from n-1 to 0, loop variable 'j'
                if 'a[j] < a[j-1]' then
                    swap 'a[j]' with 'a[j-1]'
Step5:  Run a for loop from 0 to n-1
            print 'a[i]'
Step6:  End
```

Source Code:

```c
/*Bubble Sort
  Date: 13.09.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

void main()
{
    int input_arr[20],n,i,j,temp;
    system("clear");

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

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

    for(i=0;i<n;i++)
    {
        for(j=n-1;j>0;j--)
        {
            if(input_arr[j]<input_arr[j-1])
            {
                temp=input_arr[j];
                input_arr[j]=input_arr[j-1];
                input_arr[j-1]=temp;
            }
        }
    }

    printf("Sorted array:\n");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$ ./BUBBLE
Enter maximum number of elements: 5
Enter array elements:
10 5 3 12 0
Sorted array:
0 3 5 10 12
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$
```

Insertion Sort:

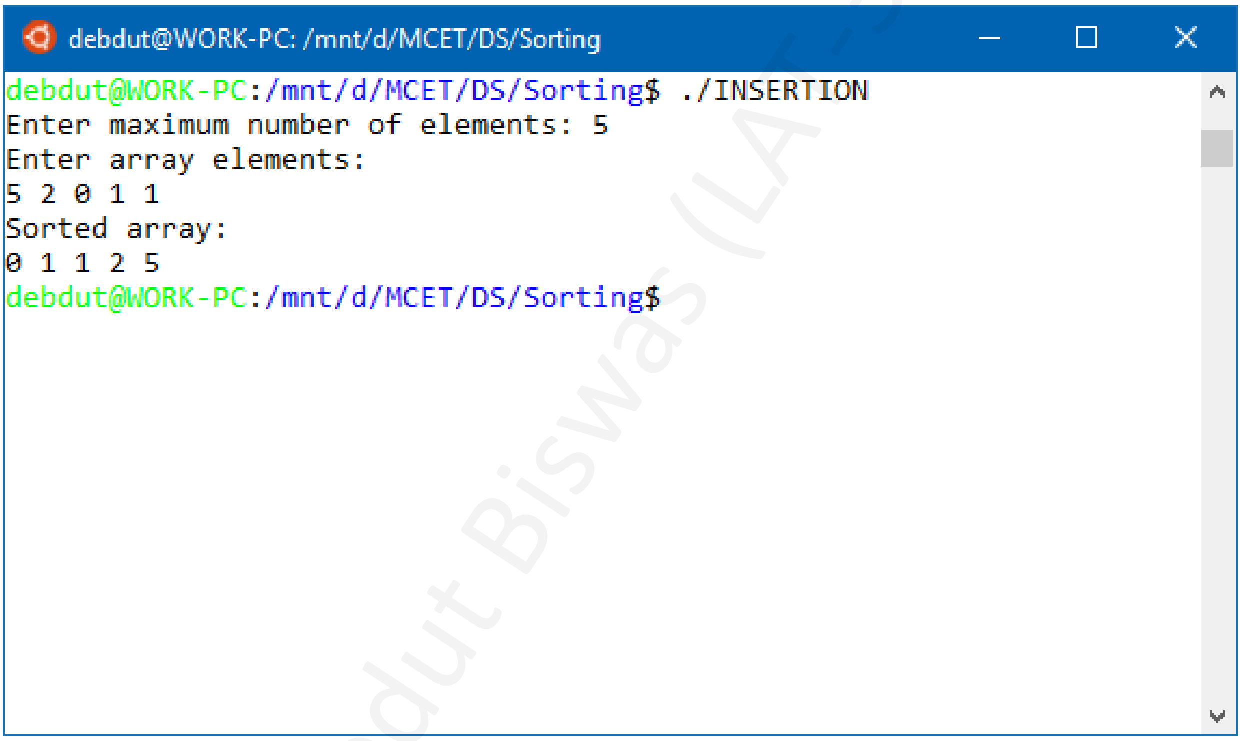
Algorithm:

```
Step1:  Start
Step2:  In main function declare an array a[n]
        'n' number of elements
        'i' and 'j' as loop variable
        'temp' to store temporary variable
Step3:  Scan 'n'
        Run a for loop from 0 to n-1
            scan a[i]
Step4:  Run a for loop from 0 to n-1, loop variable 'i'
            set 'temp=a[i]'
            run a for loop from i-1 to 0, loop variable 'j'
                if 'temp < a[j]' then
                    set 'a[j+1]=a[j]'
                else
                    break
                set 'a[j+1]=temp'
Step5:  Run a for loop from 0 to n-1
            print 'a[i]'
Step6:  End
```

Source Code:

```c
/*Insertion Sort
  Date: 20.09.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

void main()
{
    int input_arr[20],n,i,j,temp;
    system("clear");

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

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

    for(i=0;i<n;i++)
    {
        temp=input_arr[i];

        for(j=i-1;j>=0;j--)
        {
            if(temp<input_arr[j])
                input_arr[j+1]=input_arr[j];
            else
                break;
        }

        input_arr[j+1]=temp;
    }

    printf("Sorted array:\n");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}
```

Input/Output:



debdut@WORK-PC: /mnt/d/MCET/DS/Sorting

```
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$ ./INSERTION
Enter maximum number of elements: 5
Enter array elements:
5 2 0 1 1
Sorted array:
0 1 1 2 5
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$
```

Quick Sort:

Algorithm:

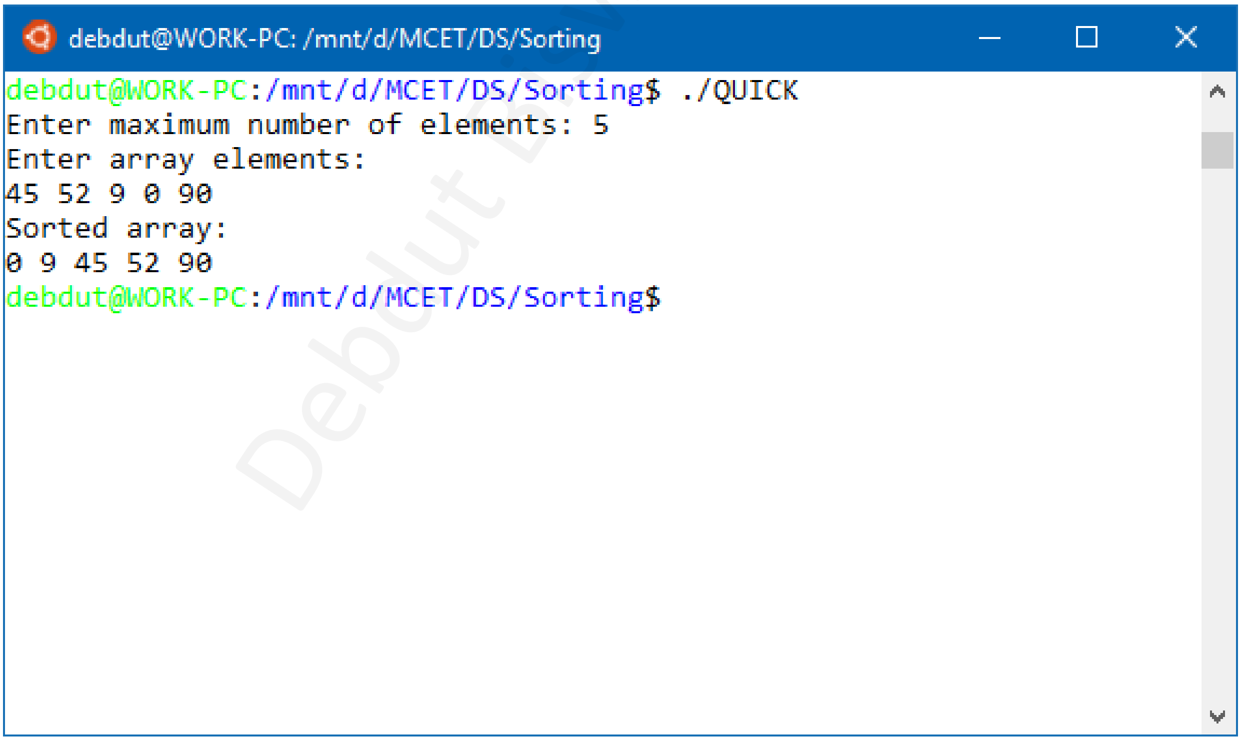
```
Step1:   Start
Step2:   'n' number of elements
         'i' and 'j' as loop variable
         'temp' for swapping variable
Step3:   In main function
             declare an array a[n]
             scan 'n' as array size
             run a for loop from 0 to n-1
                 scan 'a[i]'
             call quicksort(a,0,n-1)
             run a for loop from 0 to n-1
                 print 'a[i]'
Step4:   Define function quickSort(a[],start,end)
             if 'end>start' then
                 declare, assign and call 'pivot=partition(a,start,end)'
                 call 'quickSort(a,start,pivot-1)'  ←For pivot to left
                 call 'quickSort(a,pivot+1,end)'  ←For pivot to right
Step5:   Define function partition(a[],start,end) with return type integer
             declare and assign 'pivot=a[end]'
             set 'j=start+1'
             run a for loop from start to end-1, loop variable 'i'
                 if 'a[i] <= pivot' then
                     increment 'j' by 1
                     swap 'a[i]' and 'a[j]'
             swap 'a[j+1]' with 'a[end]'
             return 'j+1'
Step6:   End
```

Source Code:

```c
/*Quick Sort
  Date: 20.09.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

int n,i,j,temp;

void quickSort(int*,int,int);
int partition(int*,int,int);

void main()
{
    int input_arr[20];
    system("clear");

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

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

    quickSort(input_arr,0,n-1);

    printf("Sorted array:\n");
    for(i=0;i<n;i++)
        printf("%d ",input_arr[i]);

    printf("\n");
}

void quickSort(int input_arr[],int start,int end)
{
    if(end>start)
    {
        int pivot=partition(input_arr,start,end);

        quickSort(input_arr,start,pivot-1);
        quickSort(input_arr,pivot+1,end);
    }
}




//P.T.O.
```

```

int partition(int input_arr[],int start,int end)
{
    int pivot=input_arr[end];
    j=start-1;

    //send smaller elements to left of partition index
    for(i=start;i<end;i++)
    {
        if(input_arr[i]<=pivot)
        {
            j++;
            temp=input_arr[i];
            input_arr[i]=input_arr[j];
            input_arr[j]=temp;
        }
    }

    //swap partition index with pivot
    temp=input_arr[j+1];
    input_arr[j+1]=input_arr[end];
    input_arr[end]=temp;

    return(j+1);
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$ ./QUICK
Enter maximum number of elements: 5
Enter array elements:
45 52 9 0 90
Sorted array:
0 9 45 52 90
debdut@WORK-PC:/mnt/d/MCET/DS/Sorting$
```

Implement a Stack then push and pop elements of it:

Algorithm:

```
Step1:  Start
Step2:  Define 'MAX=5'
        Declare a static array 'stack[MAX]'
        Declare and assign 'top=-1'
Step3:  In main function
            Declare 'ch' as choice
            Declare 'item' as item to be pushed or popped from stack
            Run a do-while loop
            do:
                scan 'ch'
                switch case 'ch' variable
                    case 1: call 'push()'
                            break
                    case 2: set and call 'item=pop()'
                            if 'item != -100' then
                                print 'item' as deleted item
                            break
                    case 3: call 'display()'
                            break
                    default: return
            while: 'ch >= 1' and 'ch <=3' condition satisfies
Step4:  Define function push()
            Declare 'm' as new item to be pushed
            if 'top=MAX-1' then
                print 'Stack Overflow!'
                return
            scan 'm'
            increment 'top' by 1
            set 'stack[top]=m'
```

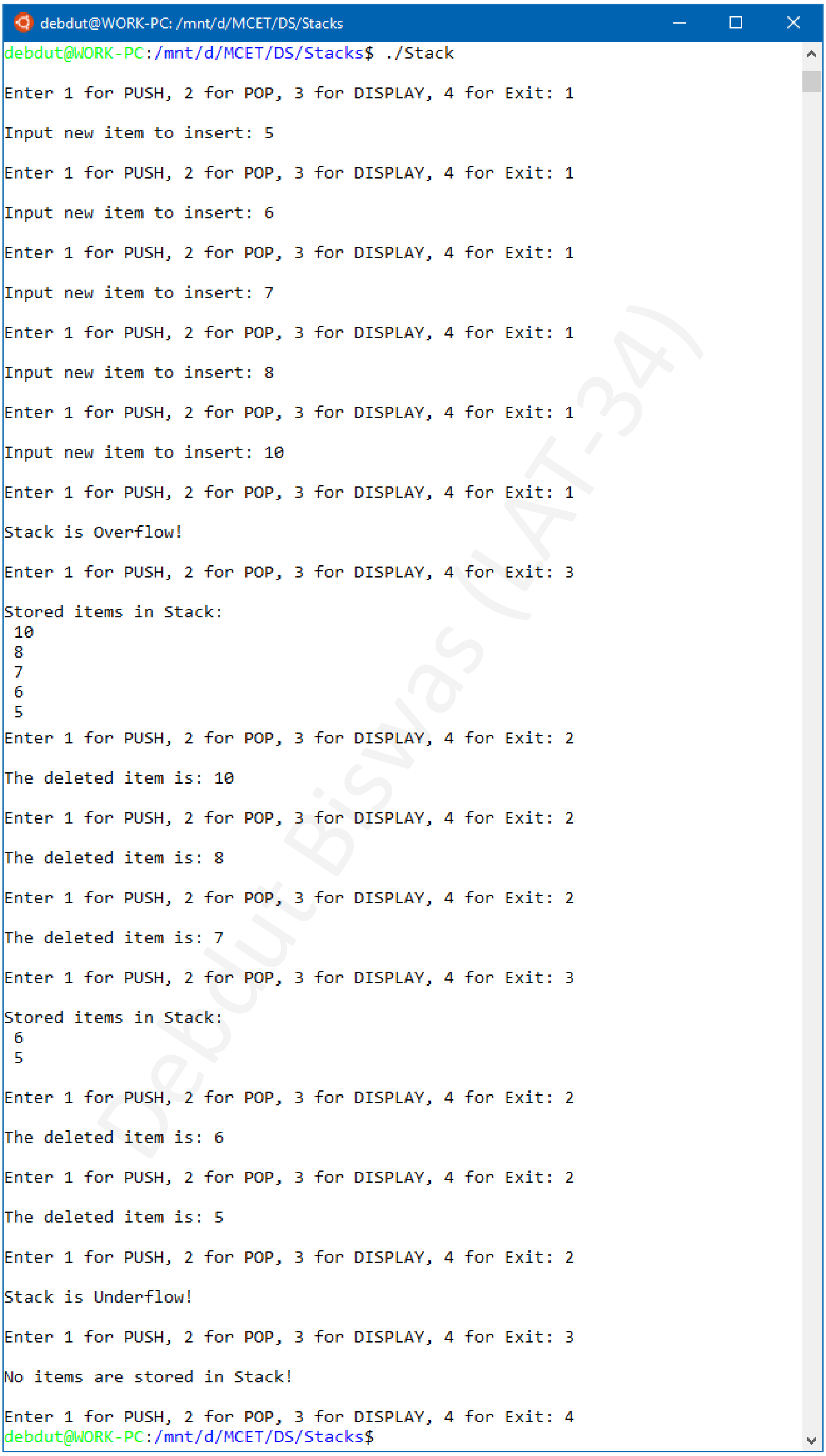
P.T.O.

```
Step5:  Define function pop() with return type integer
            Declare ‘item’ as item to be popped
            if ‘top = -1’ then
                print ‘Stack Underflow’
                return -100
            set ‘item=stack[top]’
            decrement ‘top’ by 1
            return ‘item’
Step6:  Define function display()
            Declare ‘i’ as loop variable
            if ‘top != 1’ then
                run a for loop from top to 0
                    print ‘stack[i]’
            else
                print ‘Stack is empty!’
Step7:  End
```

Source Code:

```c
/*Construct Stack and Push/Pop it
  Date: 25.10.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

#define MAX 5

int stack [MAX],top=-1;

void push();
int pop();
void display();

void main()
{
    int ch,item;
    system("clear");

    do
    {
        printf("\nEnter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                push();
                break;

            case 2:
                item=pop();
                if(item!=-100)
                {
                    printf("\nThe deleted item is: %d\n",item);
                }

                break;

            case 3:
                display();
                break;

            default:
                return;
        }  //P.T.O
    }
    while(ch>=1&&ch<=3);
}

void push() //Function for pushing items in stack.
{
    int m;

    if(top==MAX-1)
    {
        printf("\nStack is Overflow!\n");
        return;
    }

    printf("\nInput new item to insert: ");
    scanf("%d",&m);
    top++;
    stack[top]=m;
}

int pop() //Function for poping items in stack.
{
    int item;

    if(top==-1)
    {
        printf("\nStack is Underflow!\n");
        return(-100);
    }

    item=stack[top];
    top--;
    return(item);
}

void display() //Function for display items in stack.
{
    int i;
    if (top!=-1)
    {
        printf("\nStored items in Stack:\n");
        for(i=top; i>=0; i--)
        {
            printf(" %d\n",stack[i]);
        }
    }
    else
    {
        printf("\nNo items are stored in Stack!\n");
    }
}
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/Stacks$ ./Stack

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Input new item to insert: 5

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Input new item to insert: 6

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Input new item to insert: 7

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Input new item to insert: 8

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Input new item to insert: 10

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 1

Stack is Overflow!

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 3

Stored items in Stack:
 10
 8
 7
 6
 5
Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

The deleted item is: 10

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

The deleted item is: 8

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

The deleted item is: 7

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 3

Stored items in Stack:
 6
 5

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

The deleted item is: 6

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

The deleted item is: 5

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 2

Stack is Underflow!

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 3

No items are stored in Stack!

Enter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit: 4
debdut@WORK-PC:/mnt/d/MCET/DS/Stacks$
```

Construct a linear singly link list and traverse it:

Algorithm:

```
Step1:  Start
Step2:  Create a node type structure
            'info' as integer data type for storing node data
            '*link' as node type pointer for storing node type address
Step3:  Declare '*first' as node type pointer for first node
        Declare '*ptr' as node type pointer for pointer node
        Declare '*cpt' as node type pointer for current node
Step4:  In main function
            call 'create()'
            call 'traverse'
Step5:  Define function 'create()'
            allocate memory size of 'ptr'
            declare character type variable 'choice' as choice
            scan 'ptr->info'
            assign 'first=ptr' (NOTE: Coping pointer node info to first node)
            run a do-while loop
            do:
                scan 'choice'
                if 'choice' is no then
                    break
                allocate memory size of 'cpt'
                scan 'cpt->info'
                link 'ptr->link=cpt' (NOTE: linking pointer node to current node)
                set 'ptr=cpt' (NOTE: set pointer node to current node)
            while: 'choice' is yes
            set 'ptr->link=NULL' (NOTE: if it is the last node set it's pointer to NULL)
```

```
Step6:   Define function ‘traverse()’
             set ‘ptr=first’ (NOTE: set pointer node to first node)
             run a while loop when (ptr!=NULL) (NOTE: check it’s last node)
                 print ‘ptr->info’ (NOTE: print pinter node data)
                 set ‘ptr=ptr->link’ (NOTE: assign next pointer address)
```

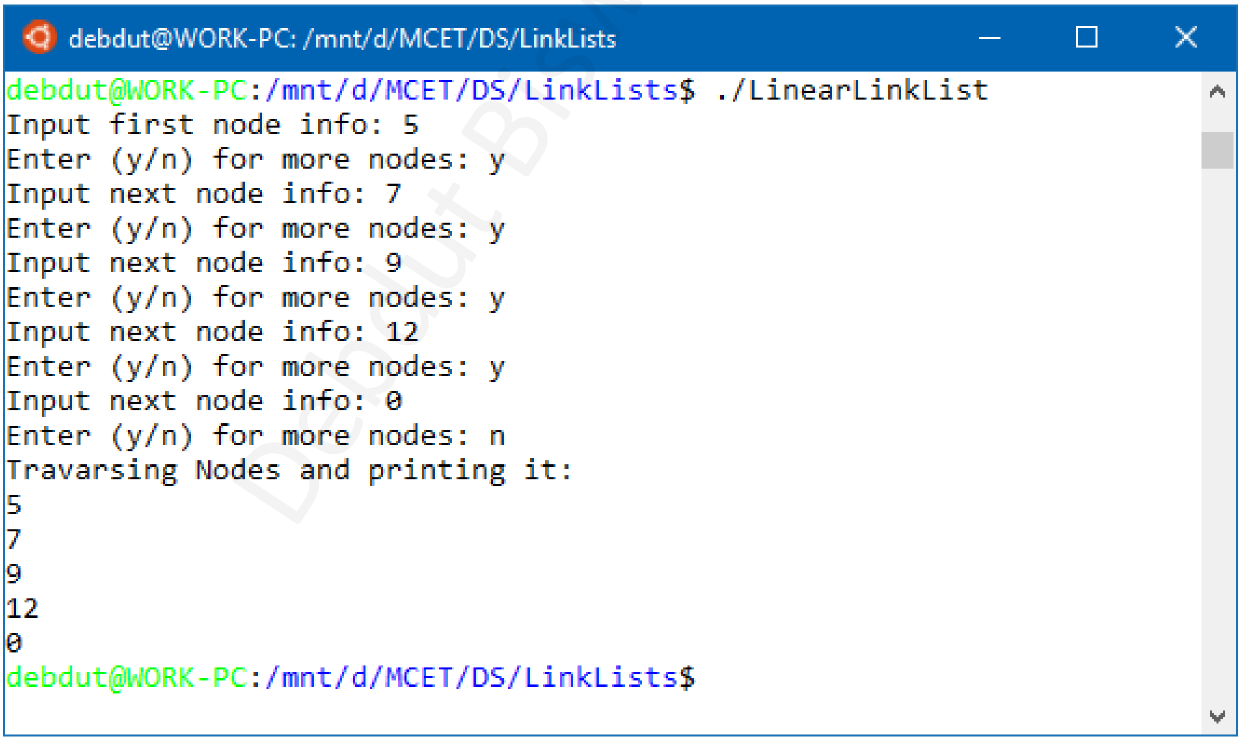
Source Code:

```c
/*Construct Linear Singly LinkList and Traverse It
  Date: 31.10.2018
  Author: Debdut
*/

#include<stdio.h>
#include<stdlib.h>

//Create structure type node
typedef struct NODE
{
    int info;
    struct NODE *link;
} node;

node *first,*ptr,*cpt;

void create();
void traverse();

void main()
{
    create();
    traverse();
}

void create()
{
    ptr=(node*)malloc(sizeof(node));

    char choice;

    printf("Input first node info: ");
    scanf("%d",&ptr->info);
    first=ptr;

    do
    {
        printf("Enter (y/n) for more nodes: ");
        scanf("%s",&choice);
        if((choice=='n') || (choice=='N'))
        {
            break;
        }

        cpt=(node*)malloc(sizeof(node));
        printf("Input next node info: ");
        scanf("%d",&cpt->info);
        ptr->link = cpt;
```

//P.T.O.

```c
            ptr = cpt;
        }
        while((choice=='y') || (choice=='Y'));

        ptr->link=NULL;
    }

    void traverse()
    {
        printf("Travarsing Nodes and printing it:\n");
        ptr=first;

        while(ptr!=NULL)
        {
            printf("%d\n",ptr->info);
            ptr=ptr->link;
        }
    }
```

Input/Output:



```
debdut@WORK-PC:/mnt/d/MCET/DS/LinkLists$ ./LinearLinkList
Input first node info: 5
Enter (y/n) for more nodes: y
Input next node info: 7
Enter (y/n) for more nodes: y
Input next node info: 9
Enter (y/n) for more nodes: y
Input next node info: 12
Enter (y/n) for more nodes: y
Input next node info: 0
Enter (y/n) for more nodes: n
Travarsing Nodes and printing it:
5
7
9
12
0
debdut@WORK-PC:/mnt/d/MCET/DS/LinkLists$
```

*ALL THE CODE REPOSITORY CAN BE FOUND IN THE LINK GIVEN BELOW:

GitHub https://github.com/DebdutBiswas/data-structures-algorithms