

▶ 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

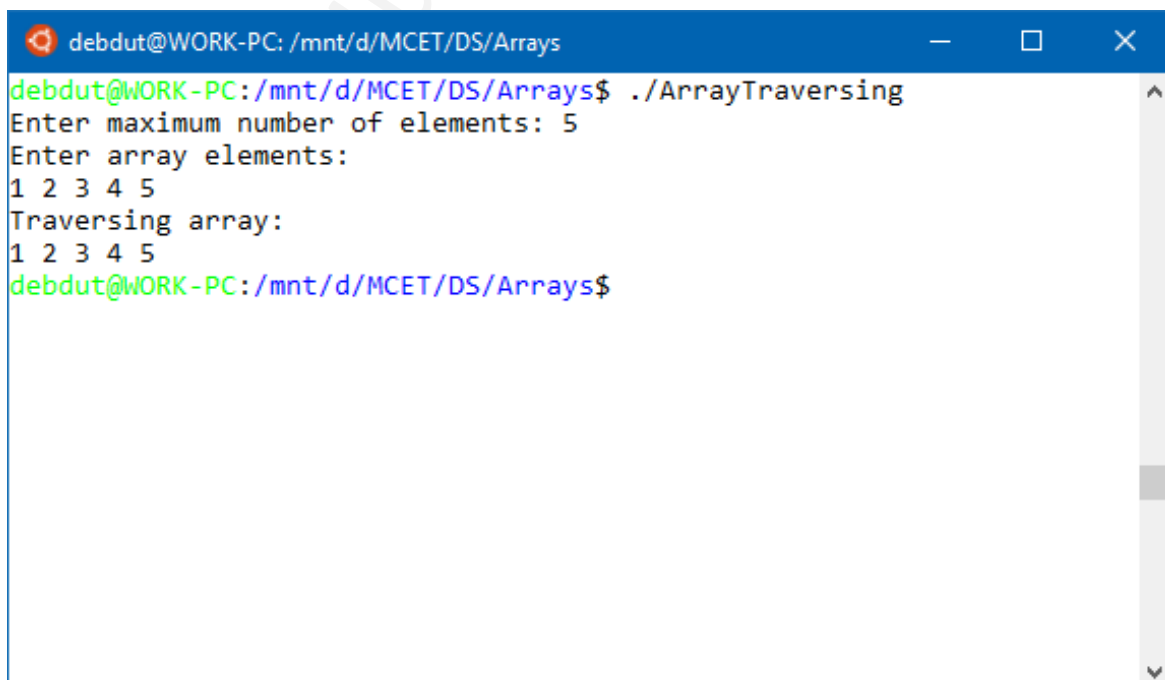
Step5: Run a for loop from 0 to n-1
Print each element

Step6: End

Source Code:

```
1. /*Traversing Static Array
2.   Date: 18.07.2018
3.   Author: Debdut
4. */
5.
6. #include<stdio.h>
7. #include<stdlib.h>
8.
9. void main()
10. {
11.     int input_arr[20],n,i;
12.     system("clear");
13.
14.     printf("Enter maximum number of elements: ");
15.     scanf("%d",&n);
16.
17.     printf("Enter array elements:\n");
18.     for(i=0;i<n;i++)
19.         scanf("%d",&input_arr[i]);
20.
21.     printf("Traversing array:\n");
22.     for(i=0;i<n;i++)
23.         printf("%d ",input_arr[i]);
24.
25.     printf("\n");
26. }
```

Input/Output:



```
debdut@WORK-PC: /mnt/d/MCET/DS/Arrays
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:
5 4 3 2 1
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:

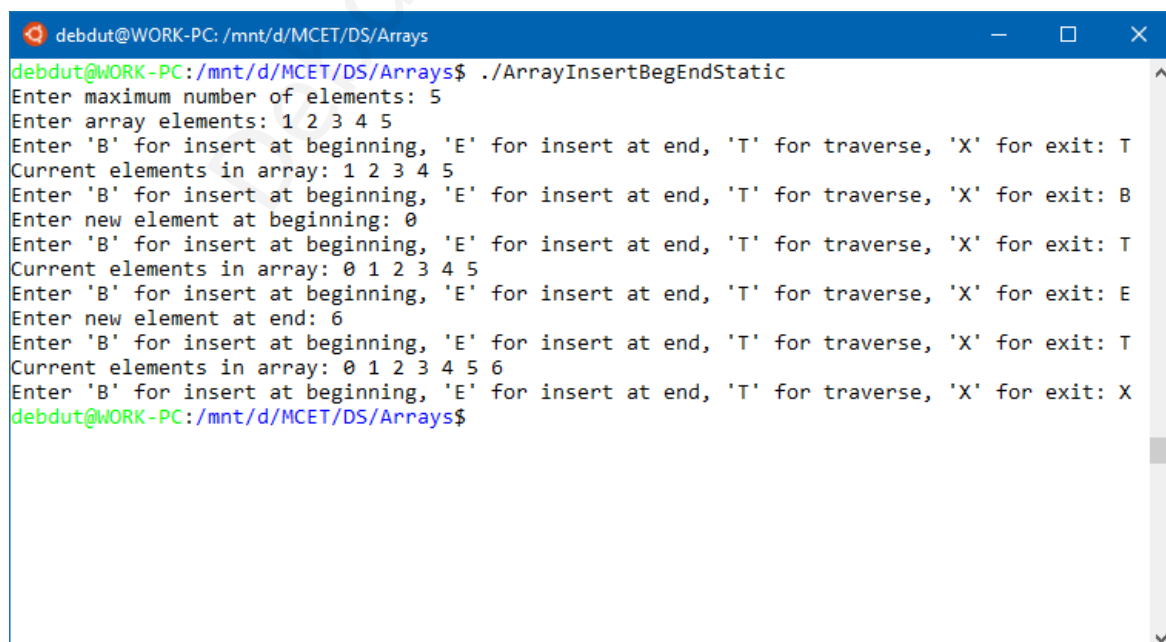
```
1.  /*Inserting Elements To Beginning and End of a Static Array
2.      Date: 26.07.2018
3.      Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i,newElement;
10.
11. void insertBeg();
12. void insertEnd();
13. void traverse();
14.
15. void main()
16. {
17.     char choice;
18.     printf("Enter maximum number of elements: ");
19.     scanf("%d",&n);
20.
21.     printf("Enter array elements: ");
22.     for(i=0;i<n;i++)
23.         scanf("%d",&input_arr[i]);
24.
25.     while(1)
26.     {
27.         printf("Enter 'B' for insert at beginning, 'E' for insert at end,
28.         'T' for traverse, 'X' for exit: ");
29.         scanf("%s",&choice);
30.
31.         switch(choice)
32.         {
33.             case 'B': insertBeg();
34.                 break;
35.             case 'E': insertEnd();
36.                 break;
37.
38.             case 'T': traverse();
39.                 break;
40.
41.             default: return;
42.         }
43.     }
44. }
45.
46.
47.
48. //P.T.O.
```

```

49.
50. void insertBeg()
51. {
52.     printf("Enter new element at beginning: ");
53.     scanf("%d",&newElement);
54.
55.     for(i=n;i>=0;i--)
56.     {
57.         input_arr[i]=input_arr[i-1];
58.     }
59.
60.     input_arr[0]=newElement;
61.     n++;
62. }
63.
64. void insertEnd()
65. {
66.     printf("Enter new element at end: ");
67.     scanf("%d",&newElement);
68.
69.     input_arr[n]=newElement;
70.     n++;
71. }
72.
73. void traverse()
74. {
75.     printf("Current elements in array: ");
76.     for(i=0;i<n;i++)
77.         printf("%d ",input_arr[i]);
78.
79.     printf("\n");
80. }

```

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:

```
1.  /*Inserting Elements To Given Position of a Static Array
2.      Date: 09.08.2018
3.      Author: Debdtut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i,elementPos,newElement;
10.
11. void insertGivenPos();
12. void traverse();
13.
14. void main()
15. {
16.     char choice;
17.     printf("Enter maximum number of elements: ");
18.     scanf("%d",&n);
19.
20.     printf("Enter array elements: ");
21.     for(i=0;i<n;i++)
22.         scanf("%d",&input_arr[i]);
23.
24.     while(1)
25.     {
26.         printf("Enter 'I' for insert at given position, 'T' for traverse,
27.         'X' for exit: ");
28.         scanf("%s",&choice);
29.
30.         switch(choice)
31.         {
32.             case 'I': insertGivenPos();
33.                 break;
34.             case 'T': traverse();
35.                 break;
36.
37.             default: return;
38.         }
39.     }
40. }
41.
42.
43.
44.
45.
46.
47.
48. //P.T.O.
```

```

49.
50. void insertGivenPos()
51. {
52.     printf("Enter the position to insert: ");
53.     scanf("%d",&elementPos);
54.     printf("Enter the new element: ");
55.     scanf("%d",&newElement);
56.
57.     for(i=n;i>=elementPos;i--)
58.     {
59.         input_arr[i]=input_arr[i-1];
60.     }
61.
62.     input_arr[elementPos]=newElement;
63.     n++;
64. }
65.
66. void traverse()
67. {
68.     printf("Current elements in array: ");
69.     for(i=0;i<n;i++)
70.         printf("%d ",input_arr[i]);
71.
72.     printf("\n");
73. }

```

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:

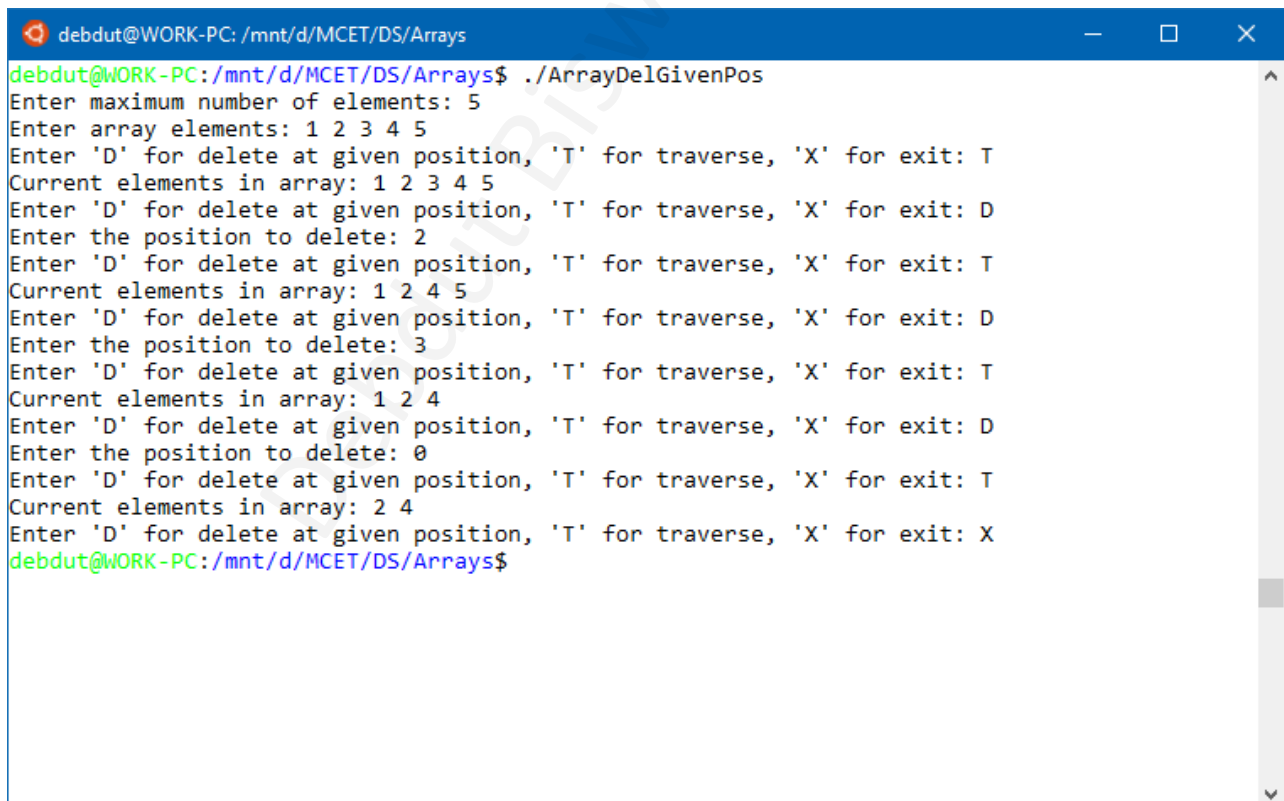
```
1.  /*Deletion Elements To Given Position of a Static Array
2.      Date: 09.08.2018
3.      Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i,elementPos;
10.
11. void deleteGivenPos();
12. void traverse();
13.
14. void main()
15. {
16.     char choice;
17.     printf("Enter maximum number of elements: ");
18.     scanf("%d",&n);
19.
20.     printf("Enter array elements: ");
21.     for(i=0;i<n;i++)
22.         scanf("%d",&input_arr[i]);
23.
24.     while(1)
25.     {
26.         printf("Enter 'D' for delete at given position, 'T' for traverse,
27.         'X' for exit: ");
28.         scanf("%s",&choice);
29.
30.         switch(choice)
31.         {
32.             case 'D': deleteGivenPos();
33.                 break;
34.             case 'T': traverse();
35.                 break;
36.
37.             default: return;
38.         }
39.     }
40. }
41.
42.
43.
44.
45.
46.
47.
48. //P.T.O.
```

```

49.
50. void deleteGivenPos()
51. {
52.     printf("Enter the position to delete: ");
53.     scanf("%d",&elementPos);
54.
55.     for(i=elementPos;i<n;i++)
56.     {
57.         input_arr[i]=input_arr[i+1];
58.     }
59.
60.     input_arr[n]=0;
61.     n--;
62. }
63.
64. void traverse()
65. {
66.     printf("Current elements in array: ");
67.     for(i=0;i<n;i++)
68.         printf("%d ",input_arr[i]);
69.
70.     printf("\n");
71. }

```

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:

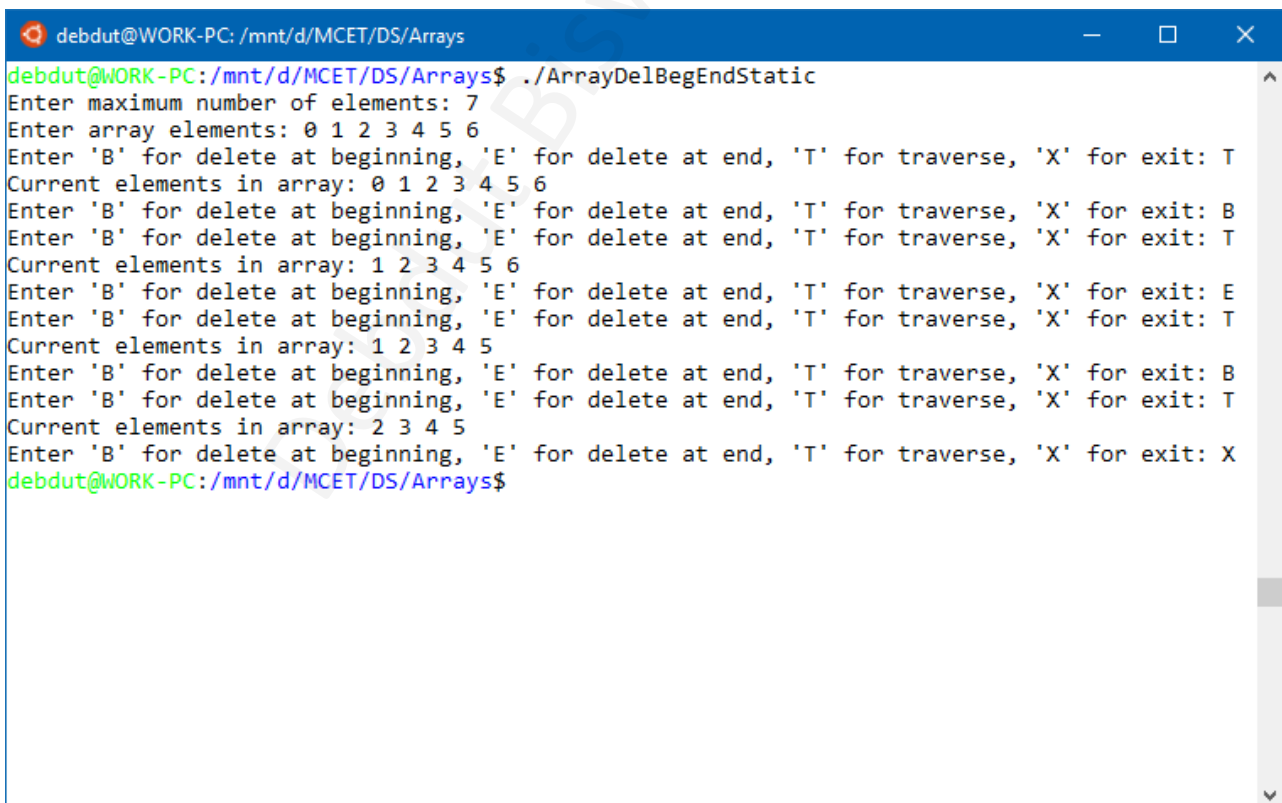
```
1.  /*Deleting Elements To Beginning and End of a Static Array
2.      Date: 09.08.2018
3.      Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i;
10.
11. void deleteBeg();
12. void deleteEnd();
13. void traverse();
14.
15. void main()
16. {
17.     char choice;
18.     printf("Enter maximum number of elements: ");
19.     scanf("%d",&n);
20.
21.     printf("Enter array elements: ");
22.     for(i=0;i<n;i++)
23.         scanf("%d",&input_arr[i]);
24.
25.     while(1)
26.     {
27.         printf("Enter 'B' for delete at beginning, 'E' for delete at end,
28.         'T' for traverse, 'X' for exit: ");
29.         scanf("%s",&choice);
30.
31.         switch(choice)
32.         {
33.             case 'B': deleteBeg();
34.                 break;
35.
36.             case 'E': deleteEnd();
37.                 break;
38.
39.             case 'T': traverse();
40.                 break;
41.
42.             default: return;
43.         }
44.     }
45.
46.
47.
48. //P.T.O.
```

```

49.
50. void deleteBeg()
51. {
52.     for(i=0;i<n;i++)
53.     {
54.         input_arr[i]=input_arr[i+1];
55.     }
56.
57.     input_arr[n]=0;
58.     n--;
59. }
60.
61. void deleteEnd()
62. {
63.     input_arr[n]=0;
64.     n--;
65. }
66.
67. void traverse()
68. {
69.     printf("Current elements in array: ");
70.     for(i=0;i<n;i++)
71.         printf("%d ",input_arr[i]);
72.
73.     printf("\n");
74. }

```

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:

```
1.  /*Linear Search
2.    Date: 30.08.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i,searchElement;
10.
11. void linearSearch();
12. void traverse();
13.
14. void main()
15. {
16.     char choice;
17.     printf("Enter maximum number of elements: ");
18.     scanf("%d",&n);
19.
20.     printf("Enter array elements: ");
21.     for(i=0;i<n;i++)
22.         scanf("%d",&input_arr[i]);
23.
24.     while(1)
25.     {
26.         printf("Enter 'S' for search element position, 'X' for exit: ");
27.
28.         scanf("%s",&choice);
29.
30.         switch(choice)
31.         {
32.             case 'S': linearSearch();
33.
34.             default: return;
35.         }
36.     }
37. }
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48. //P.T.O.
```

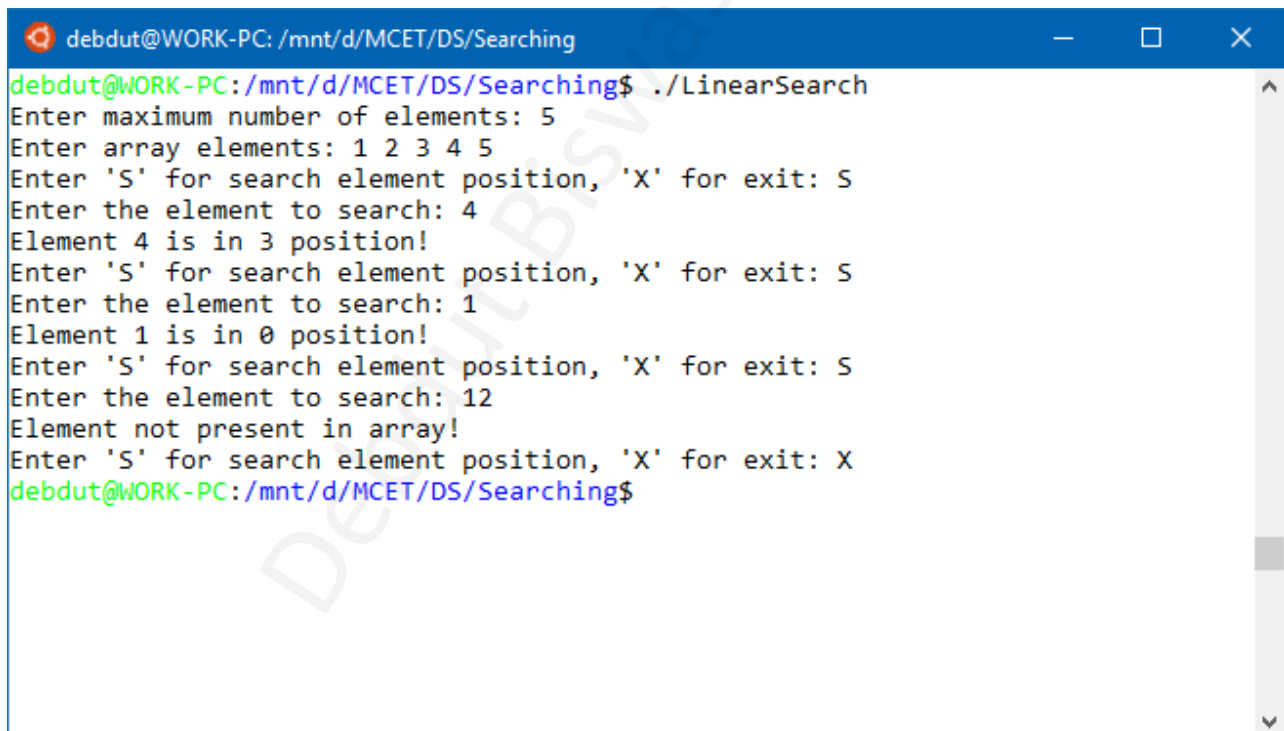


```

49.
50. void linearSearch()
51. {
52.     printf("Enter the element to search: ");
53.     scanf("%d",&searchElement);
54.
55.     for(i=0;i<n;i++)
56.     {
57.         if(input_arr[i]==searchElement)
58.         {
59.             printf("Element %d is in %d position!\n",searchElement,i);
60.             return;
61.         }
62.     }
63.     printf("Element not present in array!\n");
64. }

```

Input/Output:



```

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

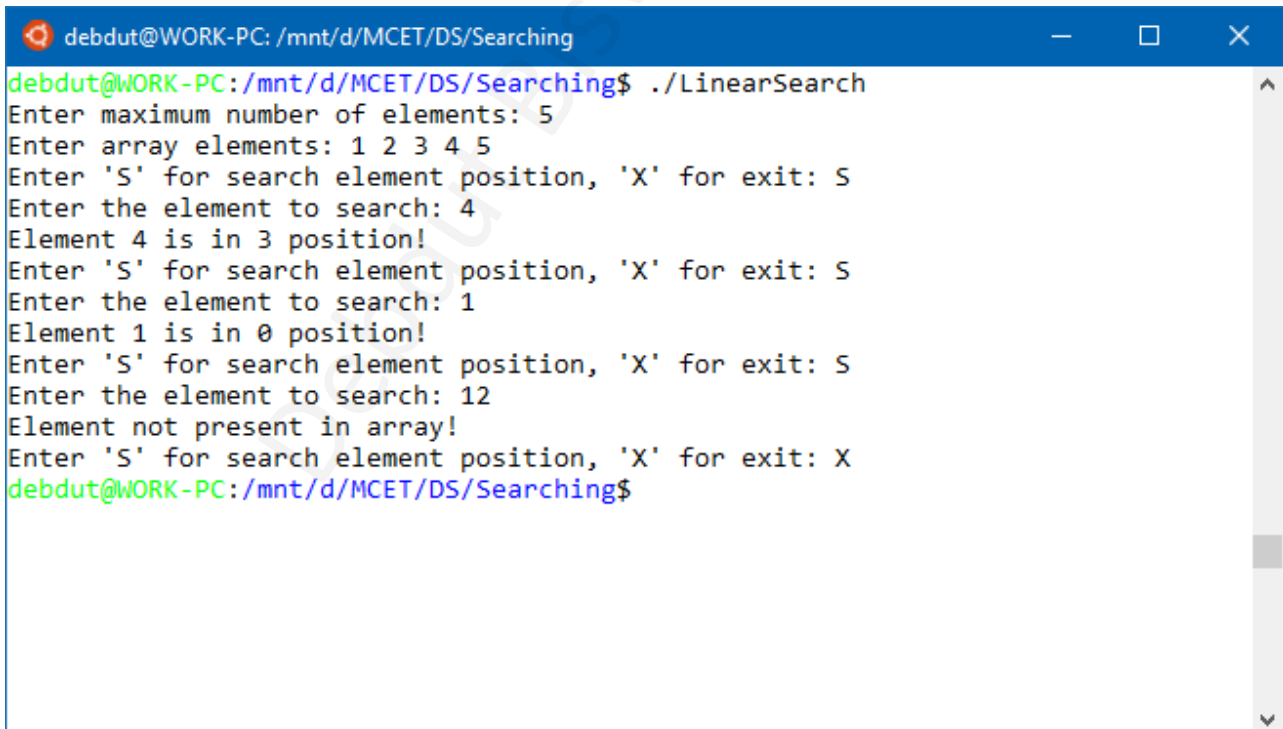
```
1.  /*Binary Search
2.    Date: 30.08.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int input_arr[20],n,i;
10.
11. void binarySearch();
12. void traverse();
13.
14. void main()
15. {
16.     char choice;
17.     printf("Enter maximum number of elements: ");
18.     scanf("%d",&n);
19.
20.     printf("Enter array elements: ");
21.     for(i=0;i<n;i++)
22.         scanf("%d",&input_arr[i]);
23.
24.     while(1)
25.     {
26.         printf("Enter 'S' for search element position, 'X' for exit: ");
27.
28.         scanf("%s",&choice);
29.
30.         switch(choice)
31.         {
32.             case 'S': binarySearch();
33.
34.             default: return;
35.         }
36.     }
37. }
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48. //P.T.O.
```

```

49.
50. void binarySearch()
51. {
52.     int left=0,right=n-1,mid,searchElement;
53.
54.     printf("Enter the element to search: ");
55.     scanf("%d",&searchElement);
56.
57.     while(left<=right)
58.     {
59.         mid=(left+right)/2;
60.
61.         if(input_arr[mid]<searchElement)
62.             left=mid+1;
63.         else if(input_arr[mid]>searchElement)
64.             right=mid-1;
65.         else
66.         {
67.             printf("Element %d is in %d position!\n",searchElement,mid);
68.             return;
69.         }
70.     }
71.     printf("Element not present in array!\n");
72. }

```

Input/Output:



```

debdut@WORK-PC: /mnt/d/MCET/DS/Searching
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 ' $a[j] < a[\text{smallest}]$ ' then
 smallest=j
 if 'smallest != i' then
 swap ' $a[\text{smallest}]$ ' with ' $a[i]$ '

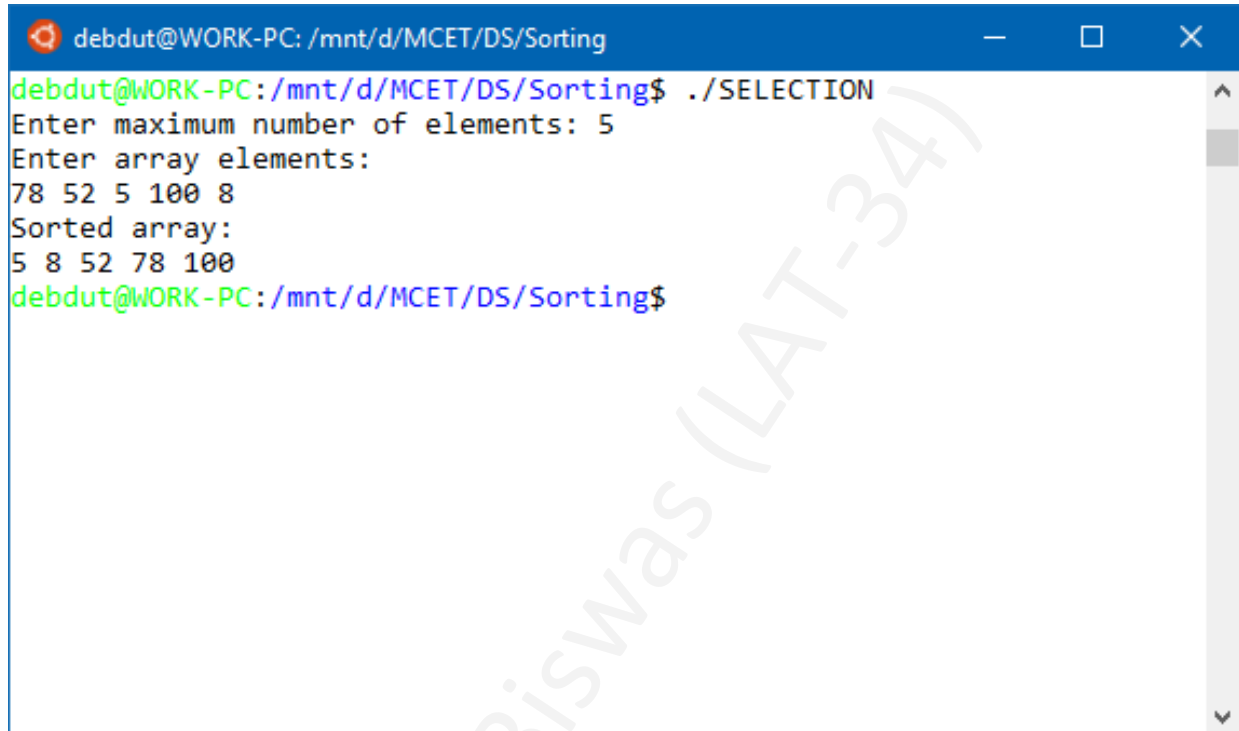
Step5: Run a for loop from 0 to $n-1$
 print ' $a[i]$ '

Step6: End

Source Code:

```
1.  /*Selection Sort
2.    Date: 13.09.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  void main()
10. {
11.     int input_arr[20],n,i,j,temp,smallest;
12.     system("clear");
13.
14.     printf("Enter maximum number of elements: ");
15.     scanf("%d",&n);
16.
17.     printf("Enter array elements:\n");
18.     for(i=0;i<n;i++)
19.         scanf("%d",&input_arr[i]);
20.
21.     for(i=0;i<n;i++)
22.     {
23.         smallest=i;
24.
25.         for(j=i+1;j<n;j++)
26.         {
27.             if(input_arr[j]<input_arr[smallest])
28.                 smallest=j;
29.         }
30.
31.         if(smallest!=i)
32.         {
33.             temp=input_arr[smallest];
34.             input_arr[smallest]=input_arr[i];
35.             input_arr[i]=temp;
36.         }
37.     }
38.
39.     printf("Sorted array:\n");
40.     for(i=0;i<n;i++)
41.         printf("%d ",input_arr[i]);
42.
43.     printf("\n");
44. }
```

Input/Output:



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

A screenshot of a terminal window titled "debdut@WORK-PC: /mnt/d/MCET/DS/Sorting". The window shows the execution of a program named "SELECTION". The user enters "5" for the maximum number of elements and "78 52 5 100 8" for the array elements. The program outputs the sorted array: "5 8 52 78 100". A large, diagonal watermark "Debdut Biswas (LAT-34)" is visible across the terminal output.

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 ' $a[j-1]$ '

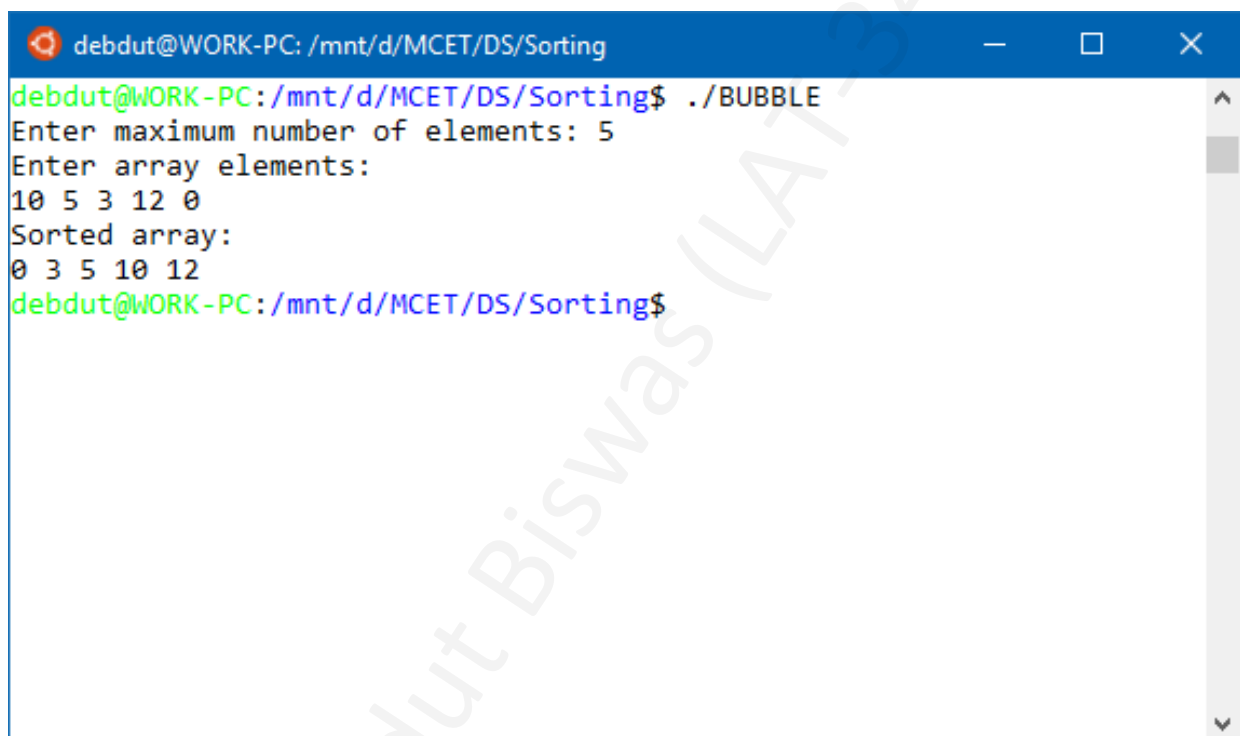
Step5: Run a for loop from 0 to $n-1$
 print ' $a[i]$ '

Step6: End

Source Code:

```
1.  /*Bubble Sort
2.    Date: 13.09.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  void main()
10. {
11.     int input_arr[20],n,i,j,temp;
12.     system("clear");
13.
14.     printf("Enter maximum number of elements: ");
15.     scanf("%d",&n);
16.
17.     printf("Enter array elements:\n");
18.     for(i=0;i<n;i++)
19.         scanf("%d",&input_arr[i]);
20.
21.     for(i=0;i<n;i++)
22.     {
23.         for(j=n-1;j>0;j--)
24.         {
25.             if(input_arr[j]<input_arr[j-1])
26.             {
27.                 temp=input_arr[j];
28.                 input_arr[j]=input_arr[j-1];
29.                 input_arr[j-1]=temp;
30.             }
31.         }
32.     }
33.
34.     printf("Sorted array:\n");
35.     for(i=0;i<n;i++)
36.         printf("%d ",input_arr[i]);
37.
38.     printf("\n");
39. }
```

Input/Output:



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

The image shows a terminal window titled "debdut@WORK-PC: /mnt/d/MCET/DS/Sorting". The user has executed the command `./BUBBLE`. The program prompts for the "maximum number of elements" (5) and "array elements" (10 5 3 12 0). It then displays the "Sorted array" (0 3 5 10 12). A large, diagonal watermark "Debdut Biswas (LATA 24)" is visible across the terminal output.

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 $\text{temp} = a[i]$
 run a for loop from $i-1$ to 0, loop variable 'j'
 if $\text{temp} < a[j]$ then
 set $a[j+1] = a[j]$
 else
 break
 set $a[j+1] = \text{temp}$

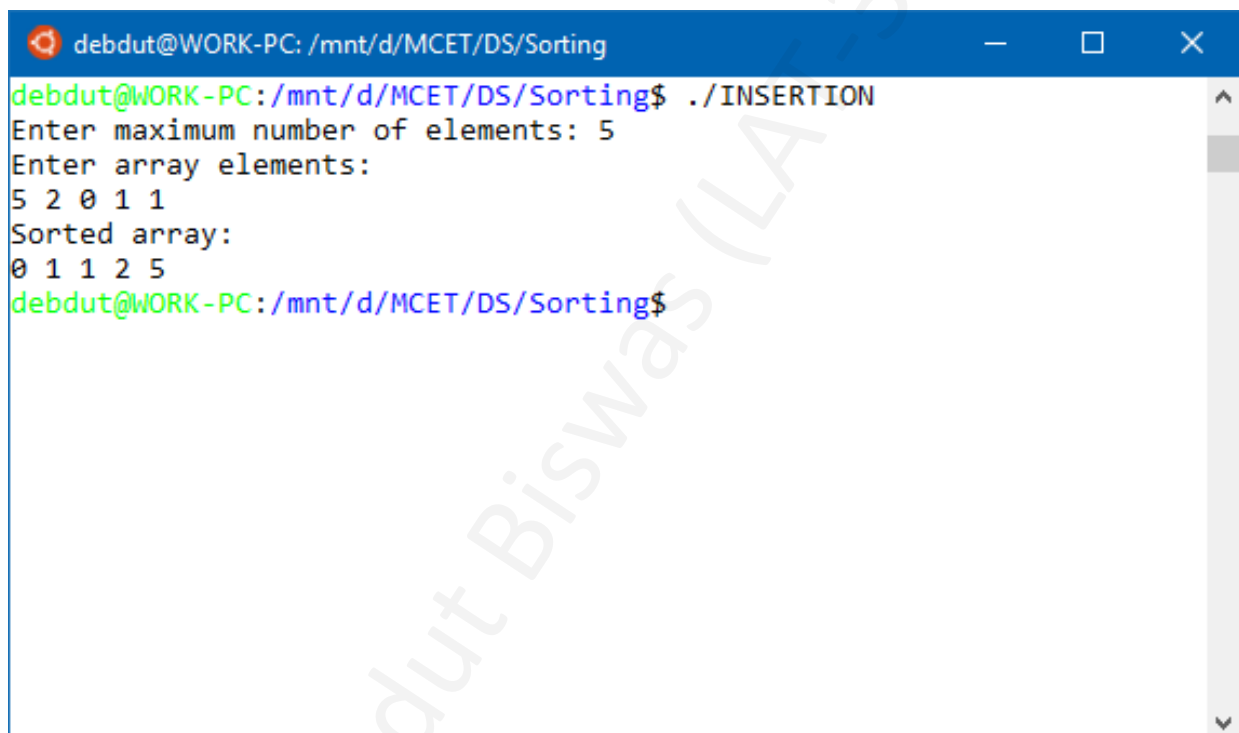
Step5: Run a for loop from 0 to $n-1$
 print $a[i]$

Step6: End

Source Code:

```
1.  /*Insertion Sort
2.    Date: 20.09.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  void main()
10. {
11.     int input_arr[20],n,i,j,temp;
12.     system("clear");
13.
14.     printf("Enter maximum number of elements: ");
15.     scanf("%d",&n);
16.
17.     printf("Enter array elements:\n");
18.     for(i=0;i<n;i++)
19.         scanf("%d",&input_arr[i]);
20.
21.     for(i=0;i<n;i++)
22.     {
23.         temp=input_arr[i];
24.
25.         for(j=i-1;j>=0;j--)
26.         {
27.             if(temp<input_arr[j])
28.                 input_arr[j+1]=input_arr[j];
29.             else
30.                 break;
31.         }
32.
33.         input_arr[j+1]=temp;
34.     }
35.
36.     printf("Sorted array:\n");
37.     for(i=0;i<n;i++)
38.         printf("%d ",input_arr[i]);
39.
40.     printf("\n");
41. }
```

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$
```

A terminal window titled "debdut@WORK-PC: /mnt/d/MCET/DS/Sorting" displays the execution of a program. The user enters the command `./INSERTION`. The program prompts for the "maximum number of elements" (5) and "array elements" (5 2 0 1 1). It then outputs the "Sorted array" (0 1 1 2 5). The terminal window has a blue title bar and standard window controls (minimize, maximize, close). A vertical scrollbar is visible on the right side of the terminal area.

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:

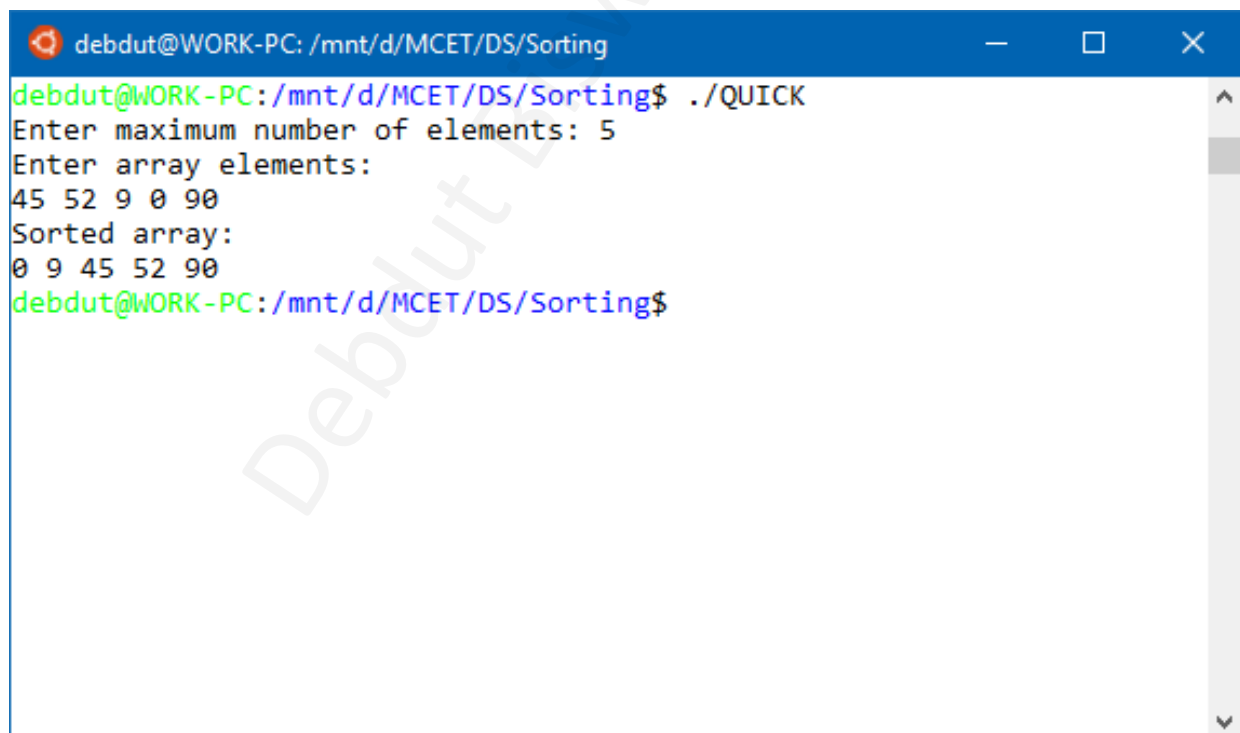
```
1.  /*Quick Sort
2.    Date: 20.09.2018
3.    Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  int n,i,j,temp;
10.
11. void quickSort(int*,int,int);
12. int partition(int*,int,int);
13.
14. void main()
15. {
16.     int input_arr[20];
17.     system("clear");
18.
19.     printf("Enter maximum number of elements: ");
20.     scanf("%d",&n);
21.
22.     printf("Enter array elements:\n");
23.     for(i=0;i<n;i++)
24.         scanf("%d",&input_arr[i]);
25.
26.     quickSort(input_arr,0,n-1);
27.
28.     printf("Sorted array:\n");
29.     for(i=0;i<n;i++)
30.         printf("%d ",input_arr[i]);
31.
32.     printf("\n");
33. }
34.
35. void quickSort(int input_arr[],int start,int end)
36. {
37.     if(end>start)
38.     {
39.         int pivot=partition(input_arr,start,end);
40.
41.         quickSort(input_arr,start,pivot-1);
42.         quickSort(input_arr,pivot+1,end);
43.     }
44. }
45.
46.
47.
48.
49. //P.T.O.
```

```

50.
51. int partition(int input_arr[],int start,int end)
52. {
53.     int pivot=input_arr[end];
54.     j=start-1;
55.
56.     //send smaller elements to left of partition index
57.     for(i=start;i<end;i++)
58.     {
59.         if(input_arr[i]<=pivot)
60.         {
61.             j++;
62.             temp=input_arr[i];
63.             input_arr[i]=input_arr[j];
64.             input_arr[j]=temp;
65.         }
66.     }
67.
68.     //swap partition index with pivot
69.     temp=input_arr[j+1];
70.     input_arr[j+1]=input_arr[end];
71.     input_arr[end]=temp;
72.
73.     return(j+1);
74. }

```

Input/Output:



```

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

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:

```
1.  /*Construct Stack and Push/Pop it
2.      Date: 25.10.2018
3.      Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  #define MAX 5
10.
11.  int stack [MAX],top=-1;
12.
13.  void push();
14.  int pop();
15.  void display();
16.
17.  void main()
18.  {
19.      int ch,item;
20.      system("clear");
21.
22.      do
23.      {
24.          printf("\nEnter 1 for PUSH, 2 for POP, 3 for DISPLAY, 4 for Exit:
25.          ");
26.          scanf("%d",&ch);
27.
28.          switch(ch)
29.          {
30.              case 1:
31.                  push();
32.                  break;
33.
34.              case 2:
35.                  item=pop();
36.                  if(item!=-100)
37.                  {
38.                      printf("\nThe deleted item is: %d\n",item);
39.                  }
40.                  break;
41.
42.              case 3:
43.                  display();
44.                  break;
45.
46.              default:
47.                  return;
48.          } //P.T.O
```

```

49.     }
50.     while(ch>=1&&ch<=3);
51. }
52.
53. void push() //Function for pushing items in stack.
54. {
55.     int m;
56.
57.     if(top==MAX-1)
58.     {
59.         printf("\nStack is Overflow!\n");
60.         return;
61.     }
62.
63.     printf("\nInput new item to insert: ");
64.     scanf("%d",&m);
65.     top++;
66.     stack[top]=m;
67. }
68.
69. int pop() //Function for popping items in stack.
70. {
71.     int item;
72.
73.     if(top== -1)
74.     {
75.         printf("\nStack is Underflow!\n");
76.         return(-100);
77.     }
78.
79.     item=stack[top];
80.     top--;
81.     return(item);
82. }
83.
84. void display() //Function for display items in stack.
85. {
86.     int i;
87.     if (top!= -1)
88.     {
89.         printf("\nStored items in Stack:\n");
90.         for(i=top; i>=0; i--)
91.         {
92.             printf(" %d\n",stack[i]);
93.         }
94.     }
95.     else
96.     {
97.         printf("\nNo items are stored in Stack!\n");
98.     }
99. }

```

Input/Output:

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


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 pointer node data)

set 'ptr=ptr->link' (NOTE: assign next pointer address)

Debdut Biswas (LAT-34)

Source Code:

```
1.  /*Construct Linear Singly LinkList and Traverse It
2.      Date: 31.10.2018
3.      Author: Debdut
4.  */
5.
6.  #include<stdio.h>
7.  #include<stdlib.h>
8.
9.  //Create structure type node
10. typedef struct NODE
11. {
12.     int info;
13.     struct NODE *link;
14. } node;
15.
16. node *first,*ptr,*cpt;
17.
18. void create();
19. void traverse();
20.
21. void main()
22. {
23.     create();
24.     traverse();
25. }
26.
27. void create()
28. {
29.     ptr=(node*)malloc(sizeof(node));
30.
31.     char choice;
32.
33.     printf("Input first node info: ");
34.     scanf("%d",&ptr->info);
35.     first=ptr;
36.
37.     do
38.     {
39.         printf("Enter (y/n) for more nodes: ");
40.         scanf("%s",&choice);
41.         if((choice=='n') || (choice=='N'))
42.         {
43.             break;
44.         }
45.
46.         cpt=(node*)malloc(sizeof(node));
47.         printf("Input next node info: ");
48.         scanf("%d",&cpt->info);
49.         ptr->link = cpt;
```

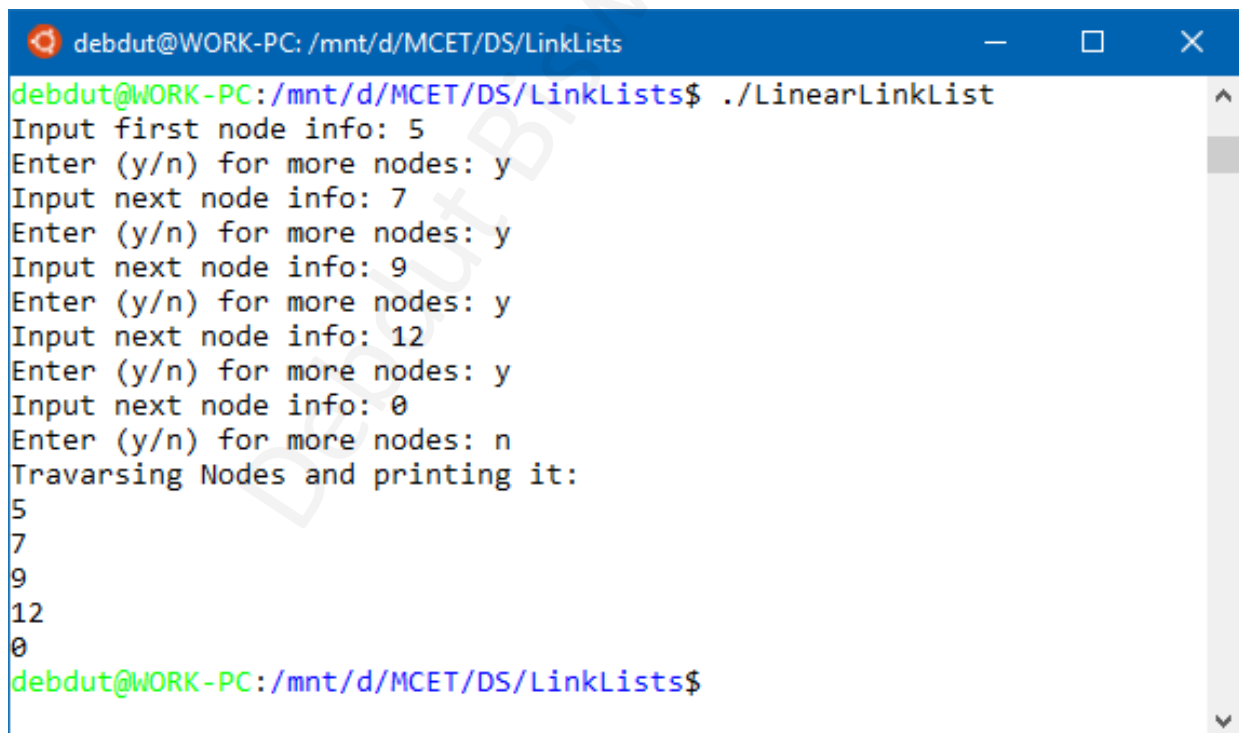
//P.T.O.


```

50.     ptr = cpt;
51. }
52. while((choice=='y') || (choice=='Y'));
53.
54.     ptr->link=NULL;
55. }
56.
57. void traverse()
58. {
59.     printf("Traversing Nodes and printing it:\n");
60.     ptr=first;
61.
62.     while(ptr!=NULL)
63.     {
64.         printf("%d\n",ptr->info);
65.         ptr=ptr->link;
66.     }
67. }

```

Input/Output:



```

debdt@WORK-PC: /mnt/d/MCET/DS/LinkLists
debdt@WORK-PC:/mnt/d/MCET/DS/LinkLists$ ./LinearLinkedList
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
Traversing Nodes and printing it:
5
7
9
12
0
debdt@WORK-PC:/mnt/d/MCET/DS/LinkLists$

```

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



<https://github.com/DebdutBiswas/data-structures-algorithms>