Assessment 1

Data structure and Algorithms

Name: Sparsh Arya
Registration Number : 17BEC0656
Slot: L39+ L40
Date: 10th December 2019

Q1. Write a menu driven program C program to swap using temporary & without temporary variable.

Pseudo Code

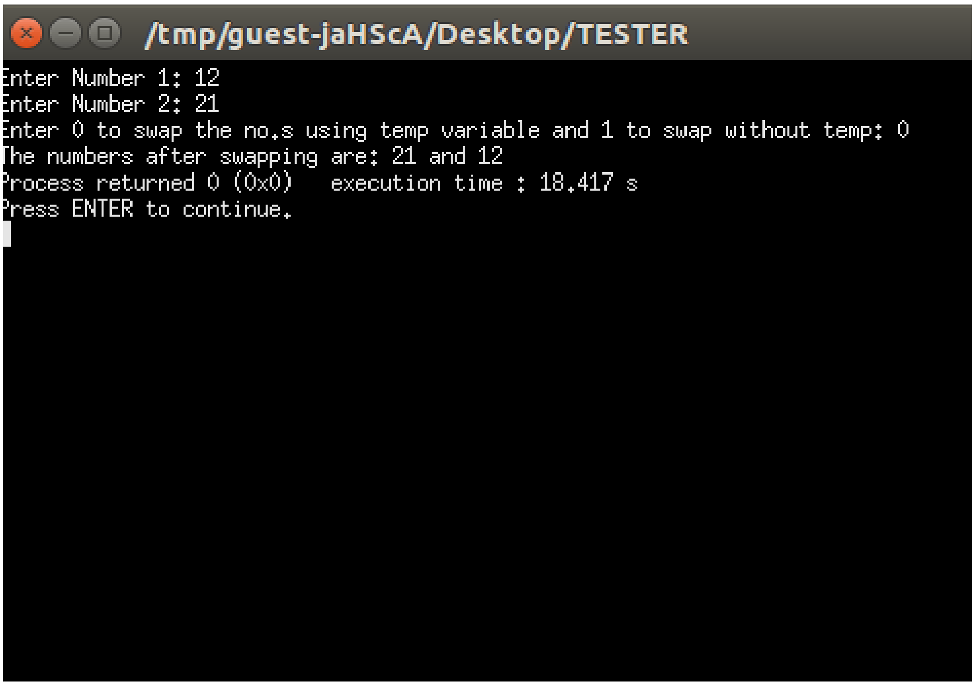
```
Input Number 1: a.
Input Number 2: b.
Input (0 or 1) for using temp or without temp variable: c

Case(c)
{

case 0: temp=a
        a=b
        b=temp
        Exit Case

case 1: a=a+b
        b=a-b
        a=a-b
        Exit Case

default: Print: Please enter a valid number.


}
Print a and b.
```

C Code

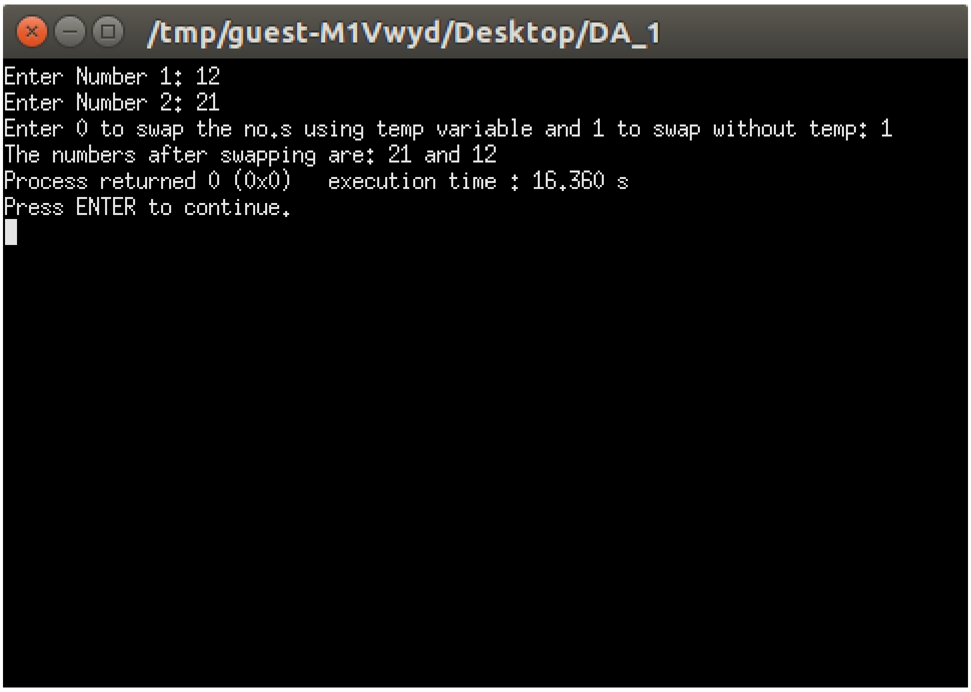
```
#include<stdio.h>
int main()
{
   int a,b,c,temp;
   printf("Enter Number 1: ");
   scanf("%d",&a);
   printf("Enter Number 2: ");
   scanf("%d",&b);
   printf("Enter 0 to swap the no.s using temp variable and 1 to swap without temp: ");
   scanf("%d",&c);
   switch(c)
   {
      case 0: temp=a;
            a=b;
            b=temp;
            break;

      case 1: a=a+b;
```

```
                b=a-b;
                a=a-b;
                break;

            default: printf("Please enter a valid number.");
        }
        printf("The numbers after swapping are: %d and %d",a,b);
        return 0;
}
```

Output



```
Enter Number 1: 12
Enter Number 2: 21
Enter 0 to swap the no.s using temp variable and 1 to swap without temp: 0
The numbers after swapping are: 21 and 12
Process returned 0 (0x0)   execution time : 18.417 s
Press ENTER to continue.
```



```
Enter Number 1: 12
Enter Number 2: 21
Enter 0 to swap the no.s using temp variable and 1 to swap without temp: 1
The numbers after swapping are: 21 and 12
Process returned 0 (0x0)   execution time : 16.360 s
Press ENTER to continue.
```

Q2. Write a menu driven C program to swap values of 3 variables with or without temp variables.

Pseudo code

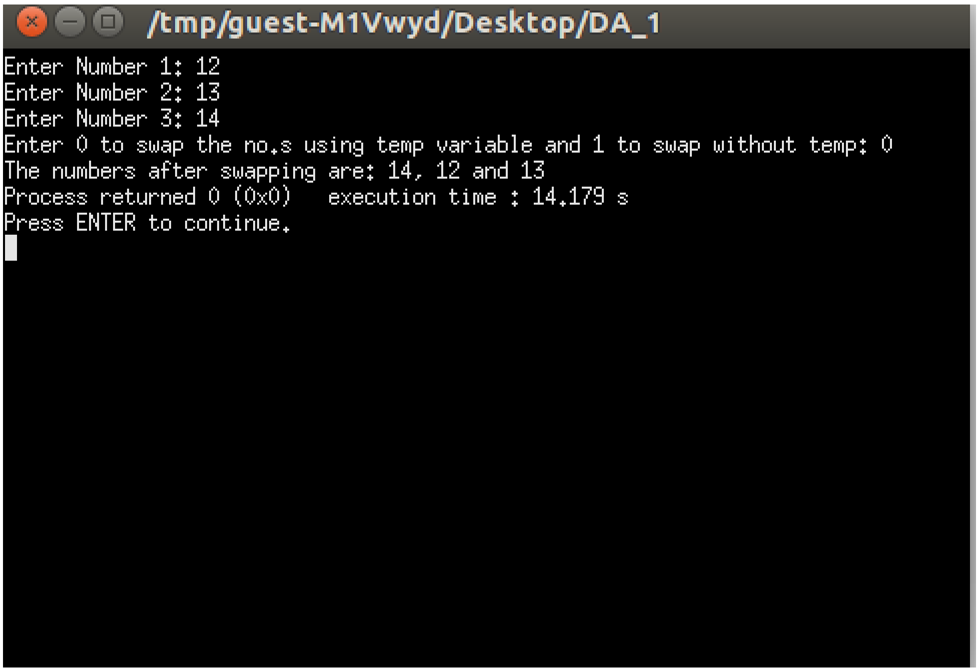
```
Input Number 1: a.
Input Number 2: b.
Input Number 3: c.
Input (0 or 1) for using temp or without temp variable: d

Case(d)
{

case 0: temp=c
        c=b
        b=a
        a=temp
        Exit Case

case 1: a=a+b+c;
        b=a-(b+c);
        c=a-(b+c);
        a=a-(b+c);
        Exit Case

default: Print: Please enter a valid number.


}
Print a, b and c.
```

C code

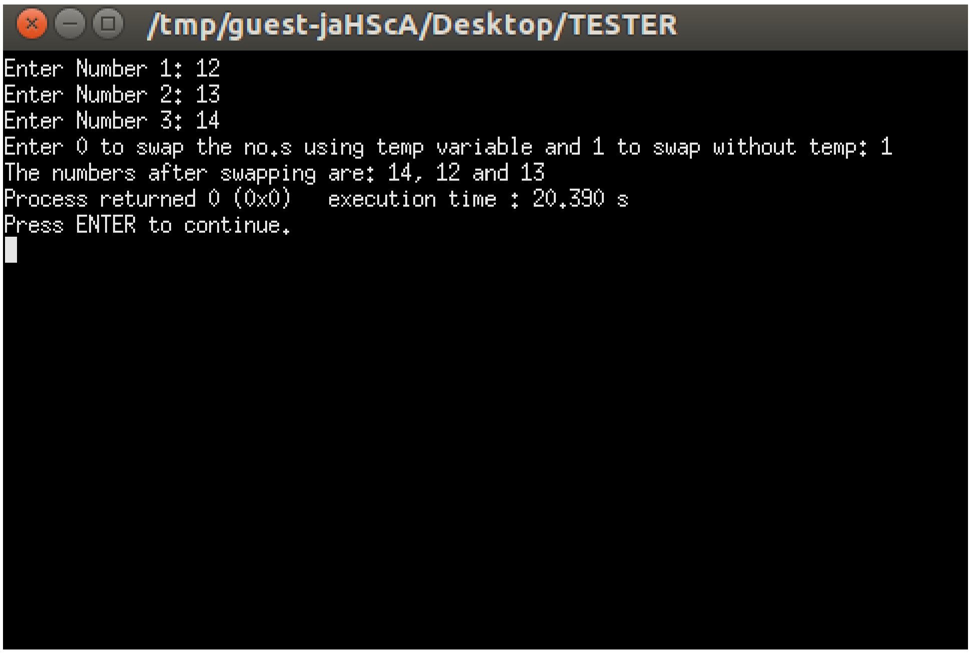
```
#include<stdio.h>
int main()
{
   int a,b,c,d,temp;
   printf("Enter Number 1: ");
   scanf("%d",&a);
   printf("Enter Number 2: ");
   scanf("%d",&b);
   printf("Enter Number 3: ");
   scanf("%d",&c);
   printf("Enter 0 to swap the no.s using temp variable and 1 to swap without temp: ");
   scanf("%d",&d);
   switch(d)
   {
      case 0: temp=c;
```

```
                c=b;
                b=a;
                a=temp;
                break;

        case 1: a=a+b+c;
                b=a-(b+c);
                c=a-(b+c);
                a=a-(b+c);
                break;

        default: printf("Please enter a valid number.");
    }
    printf("The numbers after swapping are: %d, %d and %d",a,b,c);
    return 0;

}
```

Output



```
Enter Number 1: 12
Enter Number 2: 13
Enter Number 3: 14
Enter 0 to swap the no.s using temp variable and 1 to swap without temp: 0
The numbers after swapping are: 14, 12 and 13
Process returned 0 (0x0)   execution time : 14.179 s
Press ENTER to continue.
```



```
Enter Number 1: 12
Enter Number 2: 13
Enter Number 3: 14
Enter 0 to swap the no.s using temp variable and 1 to swap without temp: 1
The numbers after swapping are: 14, 12 and 13
Process returned 0 (0x0)   execution time : 20.390 s
Press ENTER to continue.
```

Q3. Menu-driven C program to find the minimum and maximum element in an array of n numbers.

Pseudo code

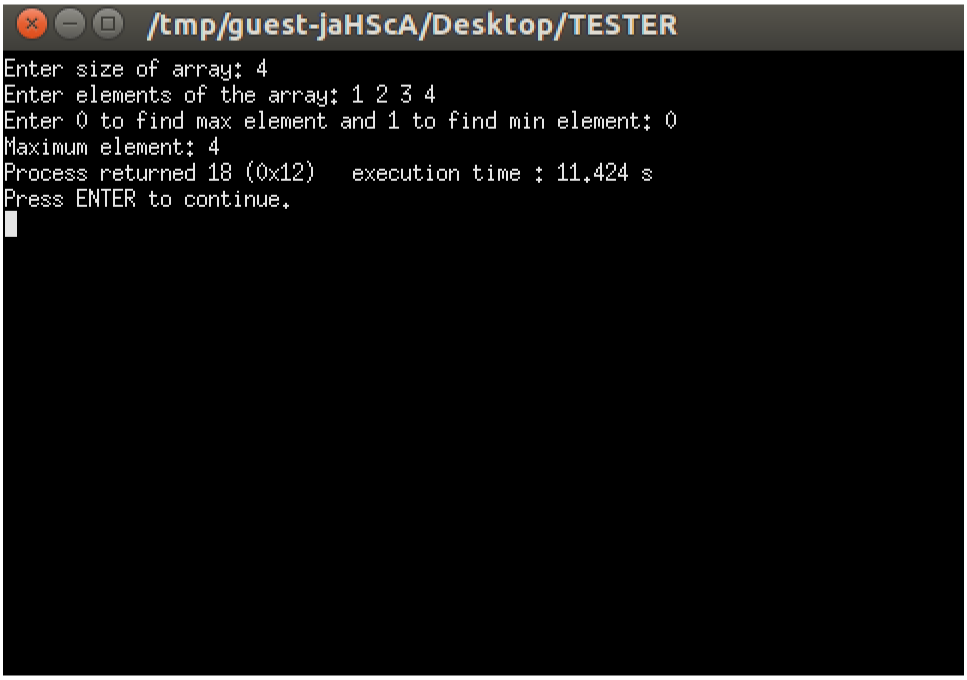
```
Input size of array: n

Loop from i=0 to n-1
        Take Input from user and store in a[i]

Input 0 for minimum element and 1 for maximum element: c

Initialize max=a[0] and min=a[0]

Case(c)
{
        Case 0:  Loop from i=0 to n-1
                        If(a[i]>max)
                                max=a[i]
                 print max
                Exit Case

        Case 1:  Loop from i=0 to n-1
                        If(a[i]<min)
                                min=a[i]
                 print min
                Exit Case

        Default: Exit case
}
```

C code

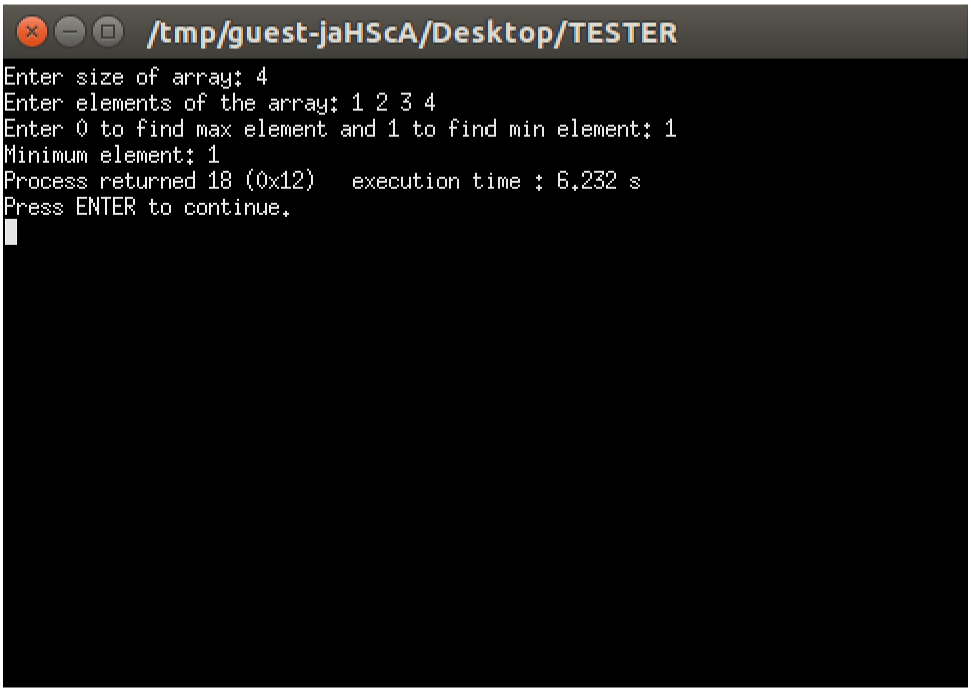
```
#include<stdio.h>
void main()
{
   int n,i,max,min,j,c;
   int a[100];
   printf("Enter size of array: ");
   scanf("%d",&n);
   printf("Enter elements of the array: ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("Enter 0 to find min element and 1 to find max element: ");
```

```
    scanf("%d",&c);
    max=a[0];
    min=a[0];
    switch(c)
    {
        case 0: for(i=0;i<n;i++)
                {
                    for(j=0;j<n;j++)
                    {
                        if(a[i]>max)
                        {
                            max=a[i];
                        }
                    }
                }
                printf("Maximum element: %d",max);
                break;

        case 1: for(i=0;i<n;i++)
                {
                    for(j=0;j<n;j++)
                    {
                        if(a[i]<min)
                        {
                            min=a[i];
                        }
                    }
                }
                printf("Minimum element: %d",min);
                break;

        default: break;

    }



}
```

Output



```
Enter size of array: 4
Enter elements of the array: 1 2 3 4
Enter 0 to find max element and 1 to find min element: 0
Maximum element: 4
Process returned 18 (0x12)   execution time : 11.424 s
Press ENTER to continue.
```



```
Enter size of array: 4
Enter elements of the array: 1 2 3 4
Enter 0 to find max element and 1 to find min element: 1
Minimum element: 1
Process returned 18 (0x12)   execution time : 6.232 s
Press ENTER to continue.
```

Q4. Menu-driven C program to search for an element using linear and binary search technique.

Pseudo Code

```
Take input for size of array: n
        Loop i=0 to n-1
                Input from user stored in a[i]
        Enter 0 for linear search and 1 for binary search: c
Case(c)
{
        case 0: Input search element
                        Loop i=0 to n-1
                                if(a[i]==search element)
                                        print search position
        case 1: b=0 // First element
                e=n // last element
                Input search element
                //Sorting
                Loop i=0 to n-1
                        Loop j=0 to n-1
                                If(a[i]>a[j])
                                        {
                                                a[i]=a[i]+a[j] //Replace values of a and b
                                                a[j]=a[i]-a[j] //Replace values of a and b
                                                a[i]=a[i]-a[j] //Replace values of a and b
                                        }
                Loop i=0 to n-1
                        Mid=(b+e)/2
                                if(s==a[mid])

                                Print element found
                                l=1 // avoid not found case
                                Exit loop

                                else if(s>a[mid])
                                  b=mid
                                else if(s<a[mid])
                                  e=mid
                        Exit loop
Default:  Print either 0 or 1.

if(l!=1)
Print element not found

}
```

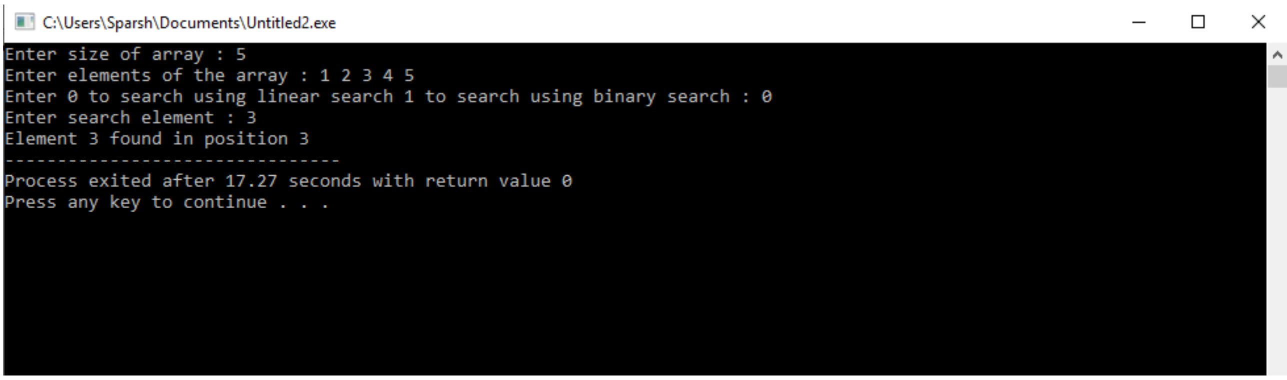
C code

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

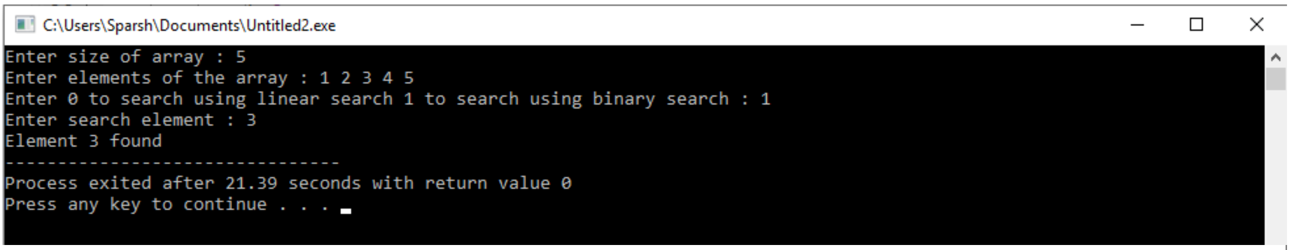
```
int main()
{
 int a[100],i,n,j,c,s,mid,e,b,l=0;
 printf("Enter size of array : ");
 scanf("%d",&n);
 printf("Enter elements of the array : ");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
 printf("Enter 0 to search using linear search 1 to search using binary search : ");
 scanf("%d",&c);
 switch(c)
 {
        case 0 : printf("Enter search element : ");
                         scanf("%d",&s);

                        for(i=0;i<n;i++)
                                if(s==a[i])
                                  {
                                        printf("Element %d found in position %d",s,i+1);
                                        l=1;
                                        break;
                                  }
                                break;
  case 1 :
                        b=0;
                        e=n;
                        printf("Enter search element : ");
                        scanf("%d",&s);
                        for(i=0;i<n;i++)
                                for(j=0;j<n;j++)
                                {
                                        if(a[i]<a[j])
                                        {
                                                a[i]=a[i]+a[j];
                                                a[j]=a[i]-a[j];
                                                a[i]=a[i]-a[j];
                                        }
                                }
                        for(i=0;i<n;i++)
                        {
                          mid=(b+e)/2;
                                if(s==a[mid])
                                  {
                                  printf("Element %d found",s);
                                  l=1;
                                  break;
                                  }
                                else if(s>a[mid])
                                  b=mid;
                                else if(s<a[mid])
                                  e=mid;
                        }
```

```
                                   break;
   default : printf("Enter either 0 or 1.");
  }
  if(l!=1)
  printf("\nElement not found\n");
  return 0;
}
```

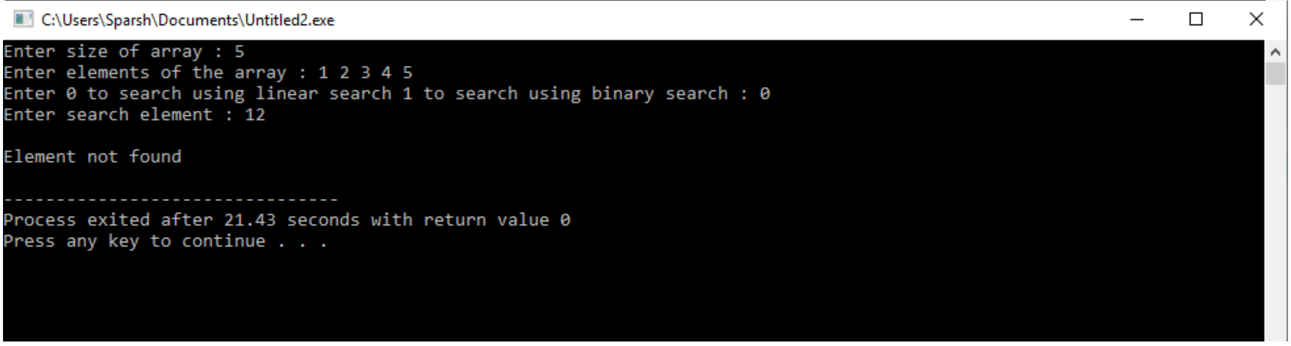
Output



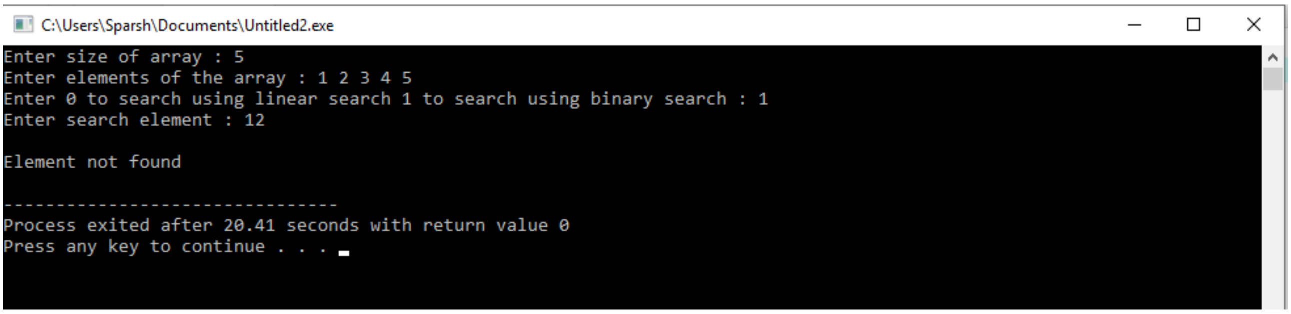
```
Enter size of array : 5
Enter elements of the array : 1 2 3 4 5
Enter 0 to search using linear search 1 to search using binary search : 0
Enter search element : 3
Element 3 found in position 3
--------------------------------
Process exited after 17.27 seconds with return value 0
Press any key to continue . . .
```



```
Enter size of array : 5
Enter elements of the array : 1 2 3 4 5
Enter 0 to search using linear search 1 to search using binary search : 1
Enter search element : 3
Element 3 found
--------------------------------
Process exited after 21.39 seconds with return value 0
Press any key to continue . . .
```



```
Enter size of array : 5
Enter elements of the array : 1 2 3 4 5
Enter 0 to search using linear search 1 to search using binary search : 0
Enter search element : 12

Element not found

--------------------------------
Process exited after 21.43 seconds with return value 0
Press any key to continue . . .
```



```
Enter size of array : 5
Enter elements of the array : 1 2 3 4 5
Enter 0 to search using linear search 1 to search using binary search : 1
Enter search element : 12

Element not found

--------------------------------
Process exited after 20.41 seconds with return value 0
Press any key to continue . . .
```

Q5. Menu-driven C program to perform insertion, deletion, display and search operations in an ordered array (ordered list).

Pseudo code

```
Do
{
        Take input for insert delete display search or exit: c
        Case(c)
        {
         Case 1: Insert element :s
                if(n==0)
                  {
                        a[0]=s;
                        n+=1;
                  }
                else if(n!=0)
                {
                  Loop from i=0 to n-1
                        {
                          if(s<a[i] && i==0)
                        {
                                        Loop from j=0 to n-1
                                        a[j+1]=a[j];
                                        a[0]=s;
                                                n=n+1;
                        }
        else if(a[i]>s && a[i-1]<s)
        {
                        Loop from j=n-1 to I decrement j
                        a[j+1]=a[j];
                        a[i]=s;
                                n=n+1;
        }
        else if(i==n-1 && a[i]<s)
        {
                a[n]=s;
        n+=1;
                }
                }
        }
        Exit case

Case 2: Take input: s
        while(i<n)
  {
        mid=(beg+end)/2;
                if(a[mid]==s)
                {
                  f=mid;break;}
                else if(a[mid]>s)
```

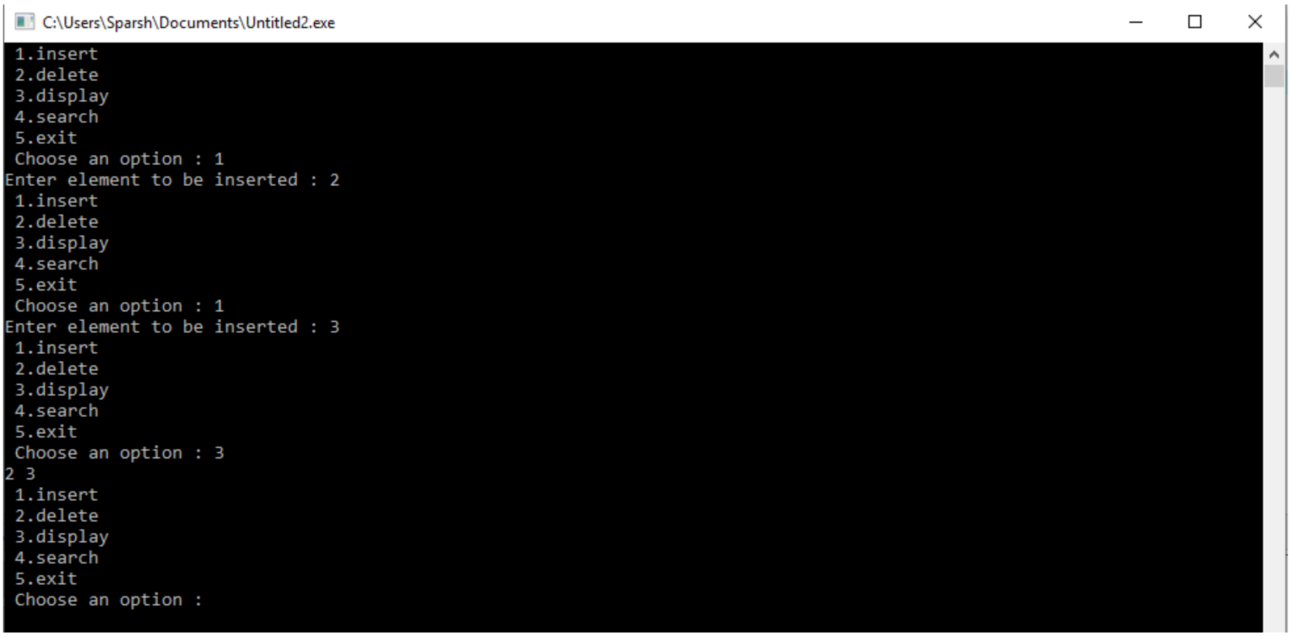
```
                        end=mid;
                else if(a[mid]<s)
                beg=mid;
  i++;
  }

Case 3: Display elements by looping the array

Case 4:   beg=0;
 end=n;
 i=0;
 f=-1;
        while(i<n)
        {
          mid=(beg+end)/2;
        if(a[mid]==s)
                {
                        f=mid;
break;
           }
        else if(a[mid]>s)
        end=mid;

  else if(a[mid]<s)
                beg=mid;
  i++;
  }
  if(f==-1)
print Element not found

  else
print Element s found in position f+1

 Exit case
Case 5:  Exit case

Default: print invalid option.
} while(c!=5);
```

C code

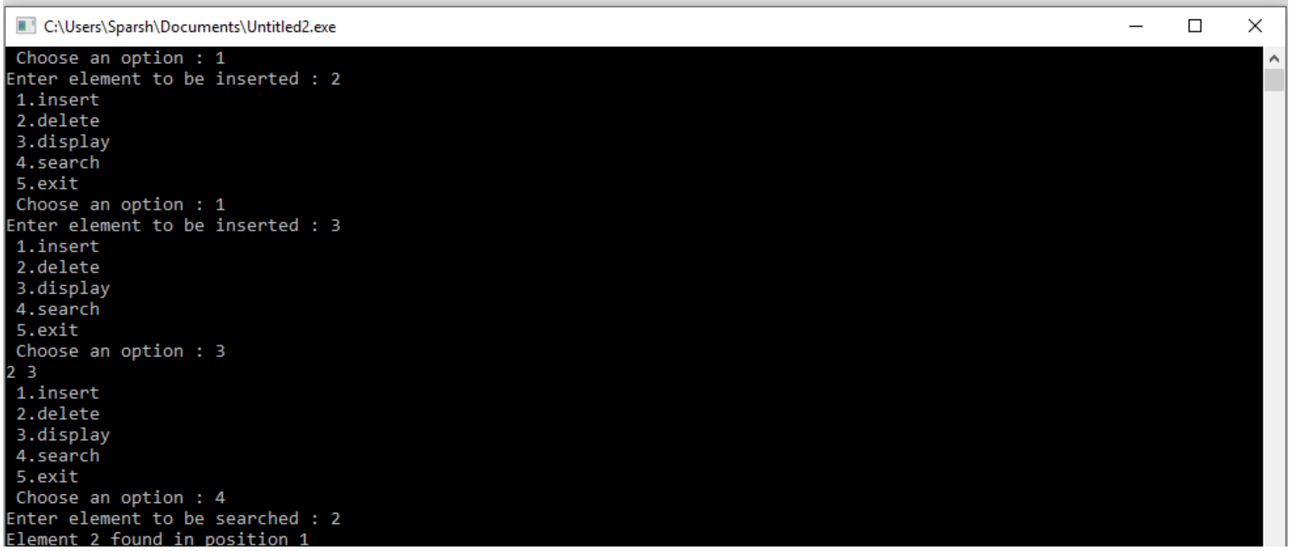
```
#include<stdio.h>
int main()
{
 int a[100],i, beg,end,mid,f,c,s,j,n=0;
f=-1;
do
 {
 printf(" 1.insert\n 2.delete\n 3.display\n 4.search\n 5.exit\n Choose an option : ");
 scanf("%d",&c);
 if(c==5)
 break;
 switch(c)
 {
case 1 : printf("Enter element to be inserted : ");
        scanf("%d",&s);
        if(n==0)
         {
            a[0]=s;
            n+=1;
         }
        else if(n!=0)
        {
         for(i=0;i<n;i++)
            {
             if(s<a[i] && i==0)
            {
                    for(j=n-1;j>=0;j--)
                    a[j+1]=a[j];
                    a[0]=s;
                        n=n+1;
            }
    else if(a[i]>s && a[i-1]<s)
    {
            for(j=n-1;j>=i;j--)
            a[j+1]=a[j];
            a[i]=s;
                n=n+1;
    }
    else if(i==n-1 && a[i]<s)
    {
        a[n]=s;
    n+=1;
        }
        }
    }
  break;

  case 2 : printf("Enter element to be deleted : ");
  scanf("%d",&s);
  beg=0;
  end=n;
  i=0;
  f=-1;
  while(i<n)
  {
    mid=(beg+end)/2;
        if(a[mid]==s)
        {
         f=mid;break;}
        else if(a[mid]>s)
```

```
                              end=mid;
                    else if(a[mid]<s)
                    beg=mid;
   i++;
   }

   if(f==-1)
   printf("Element not found\n");
   else
   {
          for(i=f;i<n-1;i++)
          a[i]=a[i+1];
          n-=1;
   }
   break;

  case 3 : for(i=0;i<n;i++)
                              printf("%d ",a[i]);
                       printf("\n");
  break;

  case 4 : printf("Enter element to be searched : ");
  scanf("%d",&s);
  beg=0;
  end=n;
  i=0;
  f=-1;
          while(i<n)
          {
            mid=(beg+end)/2;
          if(a[mid]==s)
                    {
                              f=mid;
break;
             }
          else if(a[mid]>s)
          end=mid;

   else if(a[mid]<s)
                    beg=mid;
   i++;
   }
   if(f==-1)
printf("Element not found\n");

   else
printf("Element %d found in position %d\n",s,f+1);

  break;

  case 5 : break;
  default : printf("Invalid option\n");
 }
}while(c!=5);
}
```

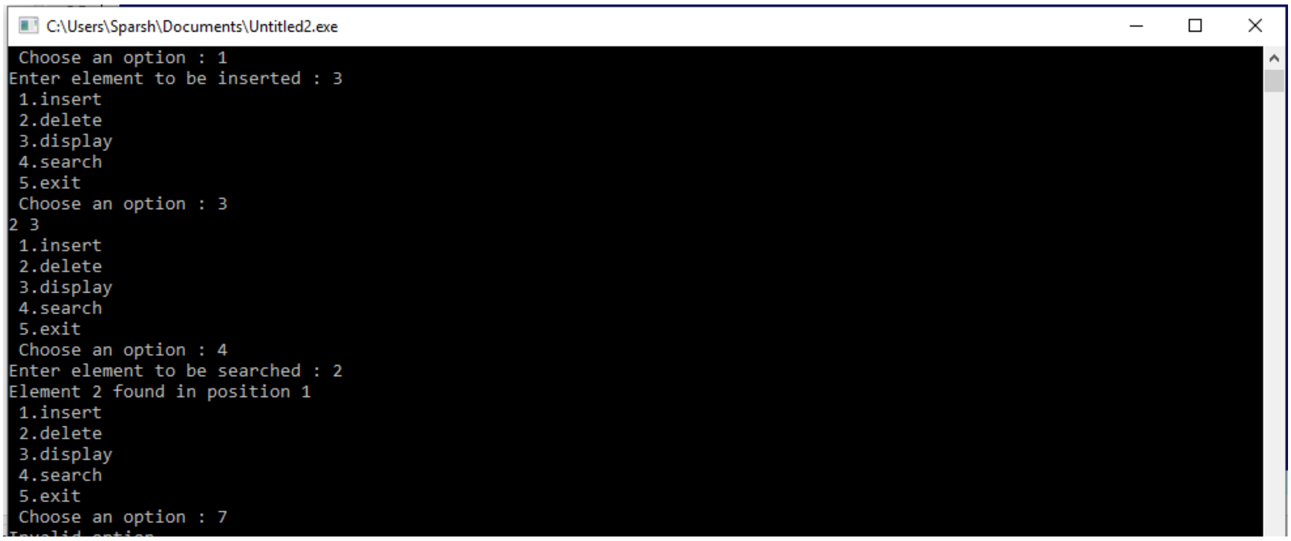
Output



```
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 1
Enter element to be inserted : 2
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 1
Enter element to be inserted : 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 3
2 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option :
```



```
 Choose an option : 1
Enter element to be inserted : 2
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 1
Enter element to be inserted : 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 3
2 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 4
Enter element to be searched : 2
Element 2 found in position 1
```



```
 Choose an option : 1
Enter element to be inserted : 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 3
2 3
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 4
Enter element to be searched : 2
Element 2 found in position 1
 1.insert
 2.delete
 3.display
 4.search
 5.exit
 Choose an option : 7
Invalid option
```

Q6. Menu-driven C program to perform insertion, deletion, display and search operations in an unordered array (unordered list).

Pseudo code

```
Do
{
        Take input for insert delete display search or exit: c
        Case(c)
        {
         Case 1: Insert element :s
                if(n==0)
                  {
                        a[0]=s;
                        n+=1;
                  }
                else if(n!=0)
                {
                  Loop from i=0 to n-1
                        {
                          if(s<a[i] && i==0)
                        {
                                        Loop from j=0 to n-1
                                        a[j+1]=a[j];
                                        a[0]=s;
                                                n=n+1;
                        }
        else if(a[i]>s && a[i-1]<s)
        {
                        Loop from j=n-1 to I decrement j
                        a[j+1]=a[j];
                        a[i]=s;
                                n=n+1;
        }
        else if(i==n-1 && a[i]<s)
        {
                a[n]=s;
        n+=1;
                }
                }
        }
        Exit case

Case 2: Take input: s
        while(i<n)
  {
        mid=(beg+end)/2;
                if(a[mid]==s)
                {
                  f=mid;break;}
                else if(a[mid]>s)
```

```
                                end=mid;
                     else if(a[mid]<s)
                     beg=mid;
   i++;
   }
```

Case 3: Display elements by looping the array

```
Case 4:   beg=0;
  end=n;
  i=0;
  f=-1;
          while(i<n)
          {
            mid=(beg+end)/2;
          if(a[mid]==s)
                     {
                                f=mid;
break;
              }
          else if(a[mid]>s)
          end=mid;

   else if(a[mid]<s)
                     beg=mid;
   i++;
   }
   if(f==-1)
print Element not found

   else
print Element s found in position f+1

  Exit case
Case 5:  Exit case

Default: print invalid option.
} while(c!=5);
```

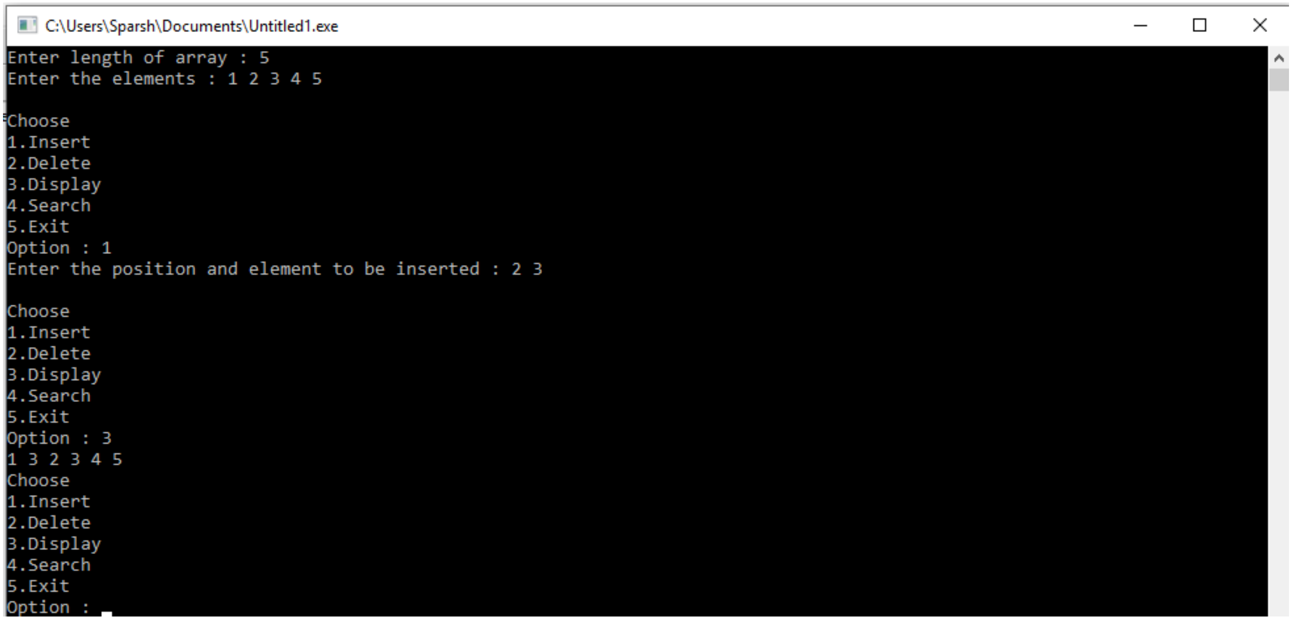
C code

```
#include<stdio.h>
 int main()
 {
  int i,a[100],n,o,e,p,c;
  printf("Enter length of array : ");
  scanf("%d",&n);
  printf("Enter the elements : ");
  for(i=0;i<n;i++)
```

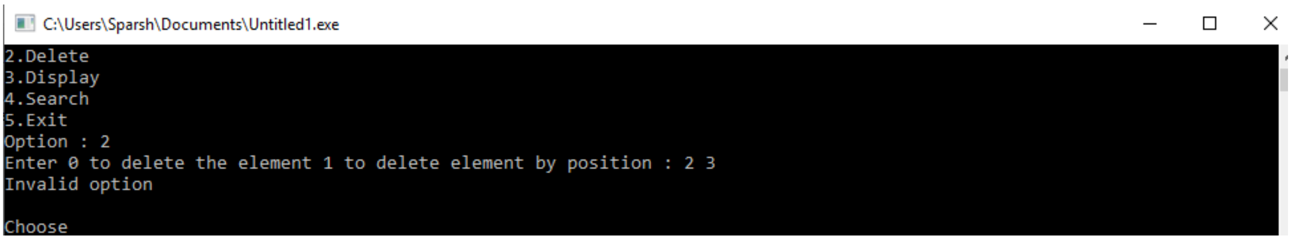
```
scanf("%d",&a[i]);
do
{
        printf("\nChoose \n1.Insert\n2.Delete\n3.Display\n4.Search\n5.Exit\nOption : ");
  scanf("%d",&c);
  switch(c)
 {
 case 1 :
  printf("Enter the position and element to be inserted : ");
  scanf("%d%d",&p,&e);
  p=p-1;
  n+=1;
  for(i=n;i>p;i--)
  a[i]=a[i-1];
  a[p]=e;
  break;

  case 2 :
  printf("Enter 0 to delete the element 1 to delete element by position : ");
  scanf("%d",&o);
  if(o==0)
  {printf("Enter element to be deleted : ");
  scanf("%d",&e);
  for(i=0;i<n;i++)
  if(a[i]==e)
        {
        p=i;
        break;
        }
  for(i=p;i<n-1;i++)
        a[i]=a[i+1];
        n-=1;
        }
  else if(o==1)
  {
                printf("Enter position of element to  be deleted : ");
        scanf("%d",&p);
        p-=1;
  for(i=p;i<n-1;i++)
  a[i]=a[i+1];
  n-=1;
        }
  else
                printf("Invalid option\n");
  break;

  case 3 :
  for(i=0;i<n;i++)
  printf("%d ",a[i]);
  break;

  case 4 :
```

```
 printf("Enter the element to be searched : ");
 scanf("%d",&e);
 for(i=0;i<n;i++)
 if(a[i]==e) printf("Element %d found in position %d\n",e,i+1);
 break;

 case 5 : break;
 default : printf("Invalid option\n");
 }
 if(c==5) break;
}while(c!=5);
}
```

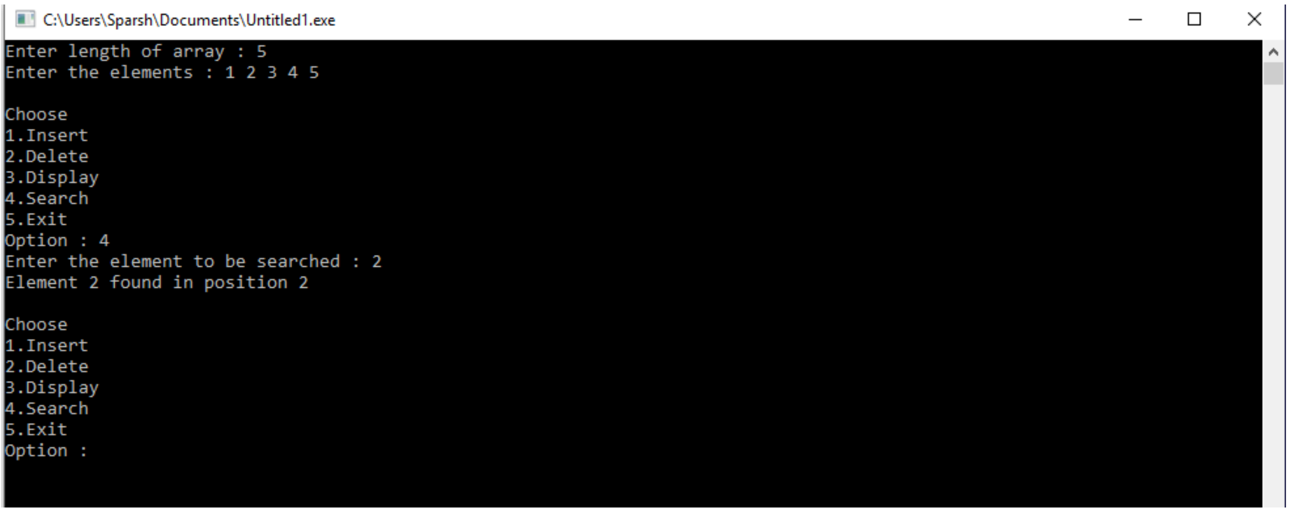
Output



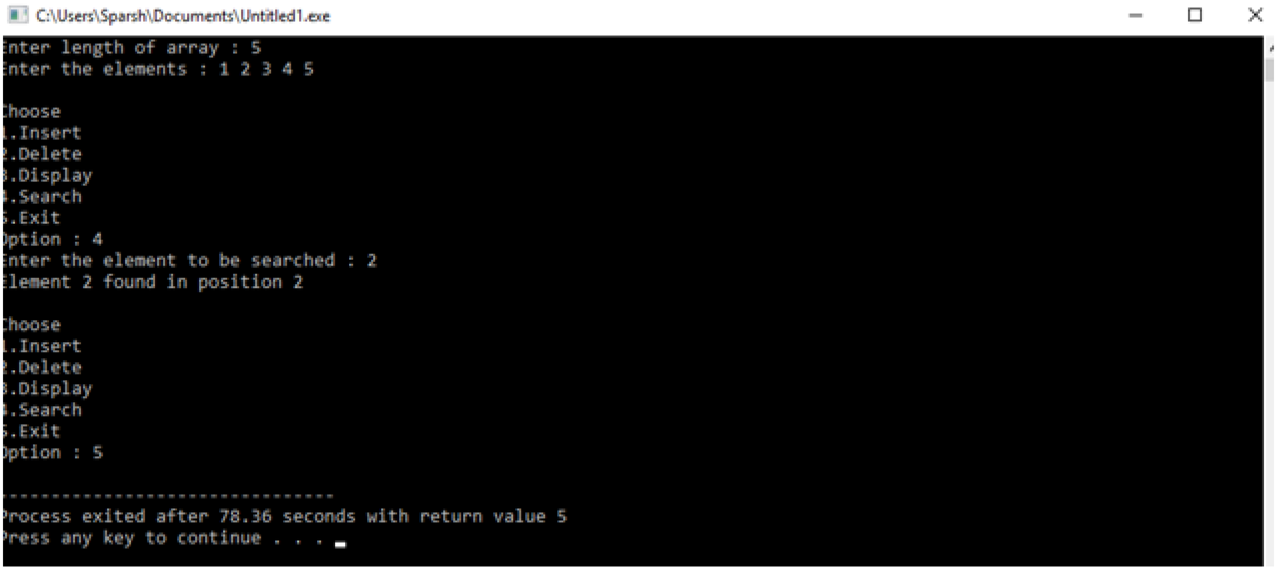
```
Enter length of array : 5
Enter the elements : 1 2 3 4 5

Choose
1.Insert
2.Delete
3.Display
4.Search
5.Exit
Option : 1
Enter the position and element to be inserted : 2 3

Choose
1.Insert
2.Delete
3.Display
4.Search
5.Exit
Option : 3
1 3 2 3 4 5
Choose
1.Insert
2.Delete
3.Display
4.Search
5.Exit
Option : _
```



```
2.Delete
3.Display
4.Search
5.Exit
Option : 2
Enter 0 to delete the element 1 to delete element by position : 2 3
Invalid option

Choose
```



```
Enter length of array : 5
Enter the elements : 1 2 3 4 5

Choose
1.Insert
2.Delete
3.Display
4.Search
5.Exit
Option : 4
Enter the element to be searched : 2
Element 2 found in position 2

Choose
1.Insert
2.Delete
3.Display
4.Search
5.Exit
Option :
```



```
nter length of array : 5
nter the elements : 1 2 3 4 5

hoose
.Insert
.Delete
.Display
.Search
.Exit
ption : 4
nter the element to be searched : 2
lement 2 found in position 2

hoose
.Insert
.Delete
.Display
.Search
.Exit
ption : 5

--------------------------------
rocess exited after 78.36 seconds with return value 5
ress any key to continue . . .
```

Q7. Menu-driven C program implement stack ADT. Perform push, pop, peek and display operations.

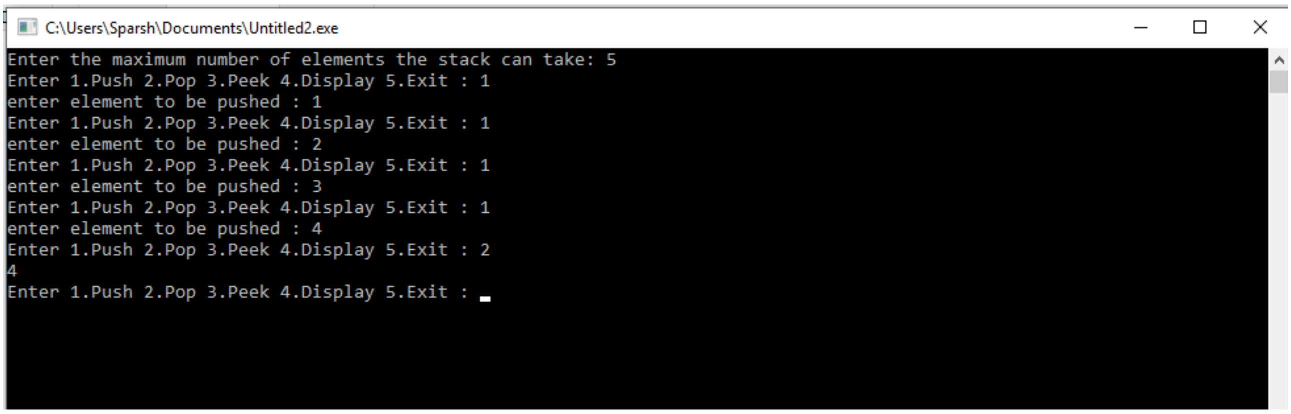
Pseudo code

```
 Take input for max of stack: max
Do
            {
                        Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit: opt
                        Case(opt)
case 1 :
                                                printf("enter element to be pushed : ");
                                                scanf("%d",&e);
  if(top==max)
  {
  Print Overflow
  Exit case
  }
  else
  {
  top++
  a[top]=e
  }
  Exit case
  case 2 : if(top==-1)
  {
  Print Underflow
  Exit case
  }
  else
  {
  Print a[top];
  top--;
  }
  break;
  case 3 :
  printf("%d\n",a[top]);
```

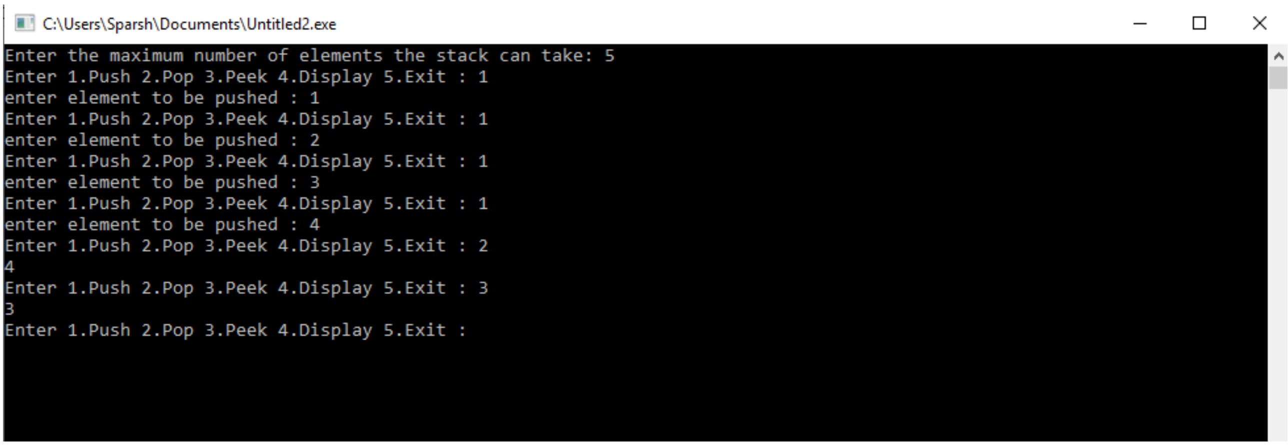
```
break;
case 4 :
loop from i=top to 0
print a[i[
        Exit case
case 5 :
Exit case
default :
Print invalid option
}while(opt!=5);
```

C code

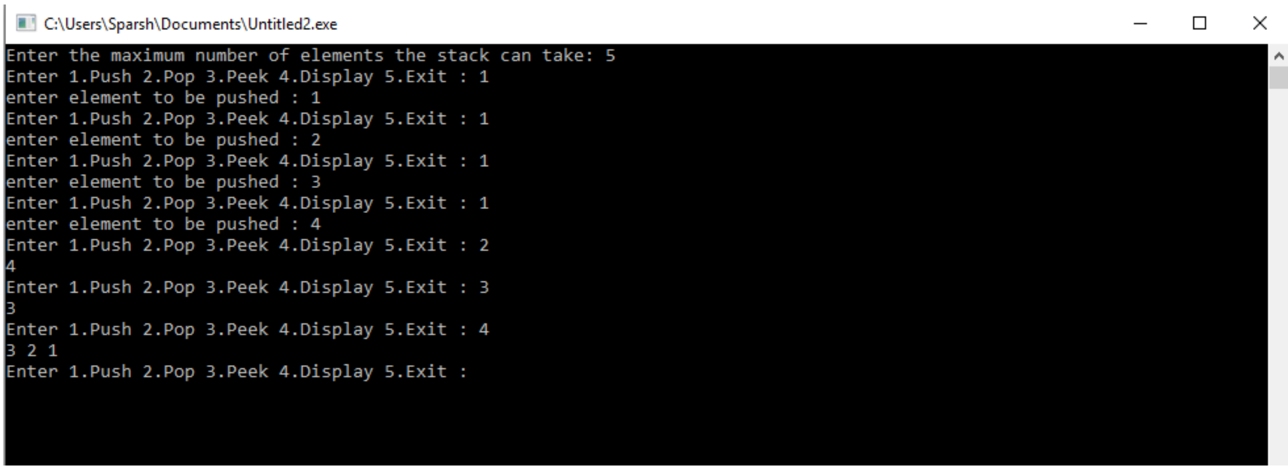
```
#include<stdio.h>
int main()
{
        int a[10],max,top=-1,opt,e,i;
        printf("Enter the maximum number of elements the stack can take: ");
        scanf("%d",&max);
         do{
                printf("Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : ");
                scanf("%d",&opt);
                switch(opt)
                {
                 case 1 :
                                printf("enter element to be pushed : ");
                                scanf("%d",&e);
 if(top==max)
 {
 printf("Overflow\n");
 break;
 }
 else
 {
 top++;
 a[top]=e;
 }
 break;
 case 2 : if(top==-1)
 {
 printf("Underflow\n");
 break;
 }
 else
 {
 printf("%d\n",a[top]);
 top--;
 }
 break;
 case 3 :
 printf("%d\n",a[top]);
 break;
 case 4 :
 for(i=top;i>=0;i--)
 printf("%d ",a[i]);
 printf("\n");
 break;
 case 5 :
 break;
 default :
 printf("Invalid option\n");
 }}while(opt!=5);
}
```



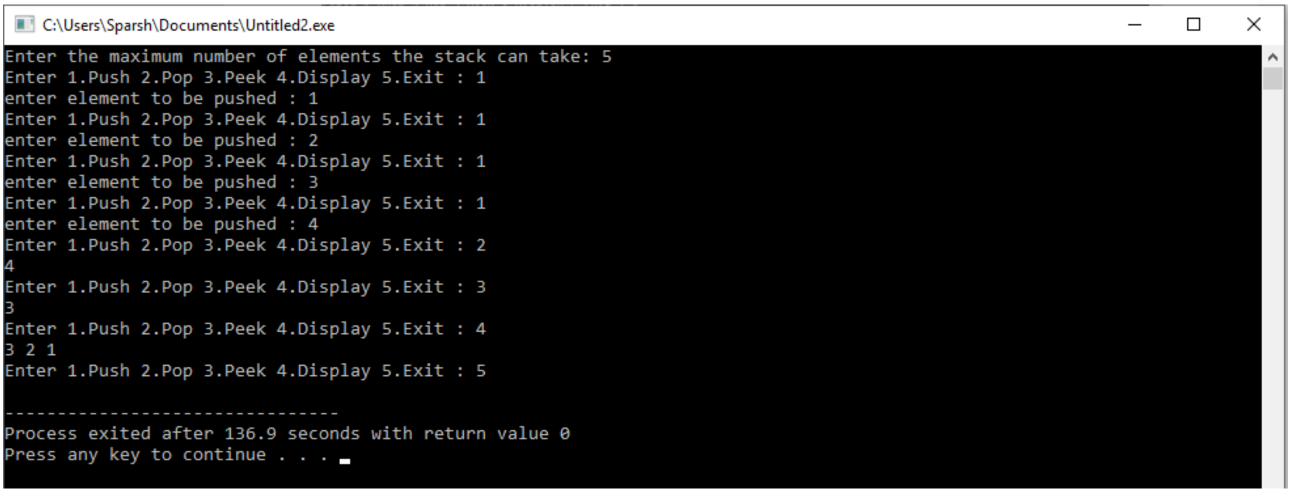
```
Enter the maximum number of elements the stack can take: 5
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 2
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 2
4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : _
```



```
Enter the maximum number of elements the stack can take: 5
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 2
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 2
4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 3
3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit :
```



```
Enter the maximum number of elements the stack can take: 5
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 2
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 2
4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 3
3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 4
3 2 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit :
```



```
Enter the maximum number of elements the stack can take: 5
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 2
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 1
enter element to be pushed : 4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 2
4
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 3
3
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 4
3 2 1
Enter 1.Push 2.Pop 3.Peek 4.Display 5.Exit : 5

--------------------------------
Process exited after 136.9 seconds with return value 0
Press any key to continue . . .
```

Q8. C program to perform expression conversion and evaluation. Take infix expression as run-time input. Convert the expression into postfix and evaluate the postfix expression.

Pseudo code

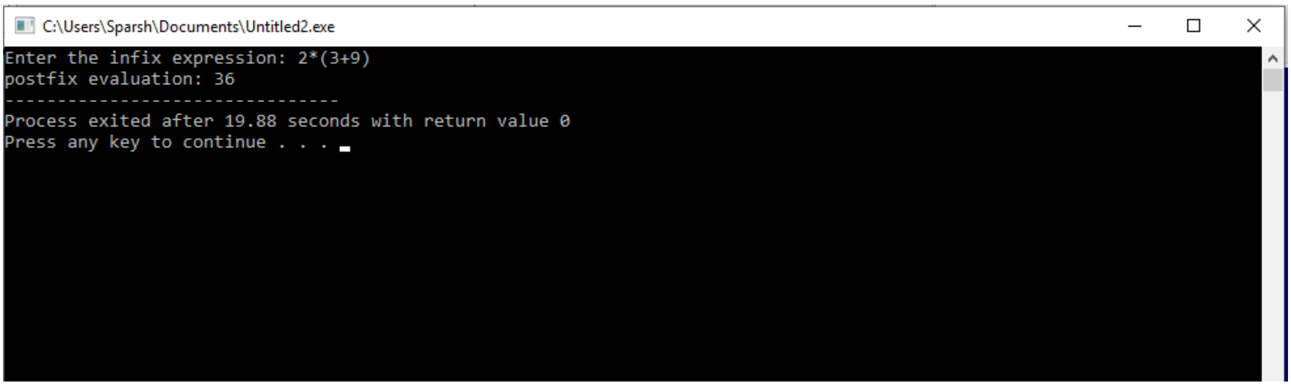
Dynamically take input for the stack
Create operations to display underflow and overflow

```
for (i = 0; exp[i]; ++i)
  {

      if (isdigit(exp[i]))
         push(stack, exp[i] - '0');


      else
      {
         int val1 = pop(stack);
         int val2 = pop(stack);
         switch (exp[i])
         {
         case '+': push(stack, val2 + val1); break;
         case '-': push(stack, val2 - val1); break;
         case '*': push(stack, val2 * val1); break;
         case '/': push(stack, val2/val1); break;
         }
      }
  }
```

C code

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include<string.h>


struct Stack
{
    int top;
    unsigned capacity;
    int* array;
};


struct Stack* createStack( unsigned capacity )
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));

    if (!stack) return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int*) malloc(stack->capacity * sizeof(int));

    if (!stack->array) return NULL;

    return stack;
}

int isEmpty(struct Stack* stack)
{
    return stack->top == -1 ;
}

char peek(struct Stack* stack)
{
    return stack->array[stack->top];
}

char pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--] ;
    return '$';
}

void push(struct Stack* stack, char op)
{
    stack->array[++stack->top] = op;
}



int evaluatePostfix(char* exp)
{

    struct Stack* stack = createStack(strlen(exp));
    int i;


    if (!stack) return -1;


    for (i = 0; exp[i]; ++i)
    {

        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');


        else
        {
            int val1 = pop(stack);
            int val2 = pop(stack);
            switch (exp[i])
            {
            case '+': push(stack, val2 + val1); break;
            case '-': push(stack, val2 - val1); break;
            case '*': push(stack, val2 * val1); break;
            case '/': push(stack, val2/val1); break;
            }
        }
    }
    return pop(stack);
}


int main()
{
    char exp[20];
            printf("Enter the infix expression: ");
    scanf("%[^\n]",&exp);
    printf ("postfix evaluation: %d", evaluatePostfix(exp));
    return 0;
}
```

Output



```
Enter the infix expression: 2*(3+9)
postfix evaluation: 36
--------------------------------
Process exited after 19.88 seconds with return value 0
Press any key to continue . . .
```