Assessment-1: Structures-Arrays-Stack-Queue

Reg.no: 20BDS0117

Name: SANJANA.SAIRAMA

Slot: L37+L38

(1) Create a registration form application by taking the details like username, address, phone number, email along with password and confirm password (should be same as password).Ensure that the password is of 8 characters with only numbers and alphabets. Take such details for 5 users and display the details. In place of password display “****”. (Use Structures).

Pseudo code:

```
1.Begin
2.Initialize structure information
3. procedure takePassword()
   while(1)
        ch = getch()
        if(ch == 13)
             break
        endif
         else if(ch == 8)
              password[--i] = '\0'
        endelseif
        else if(ch == 32 || ch == 9)
              continue
        endelseif
        else
         if((ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9') || (ch >= 'A' && ch <= 'Z'))
             password[i++] = ch
              print "*"
        endif
        otherwise print invalid format
```

```
 if(i != 8)
       print Invalid password format!
       Print Please Enter again: "
       takePassword()
endif
otherwise password[i++] = '\0'
4.Take input from user for all the given details
5. if(strcmp(password, arr[i].password) != 0)
           Print The confirm password is incorrect, please enter info again!!
           i—
   endif
    otherwise strcpy(arr[i].confirmPass, password)
6.Print all the details
7.End
```

<u>Code:</u>

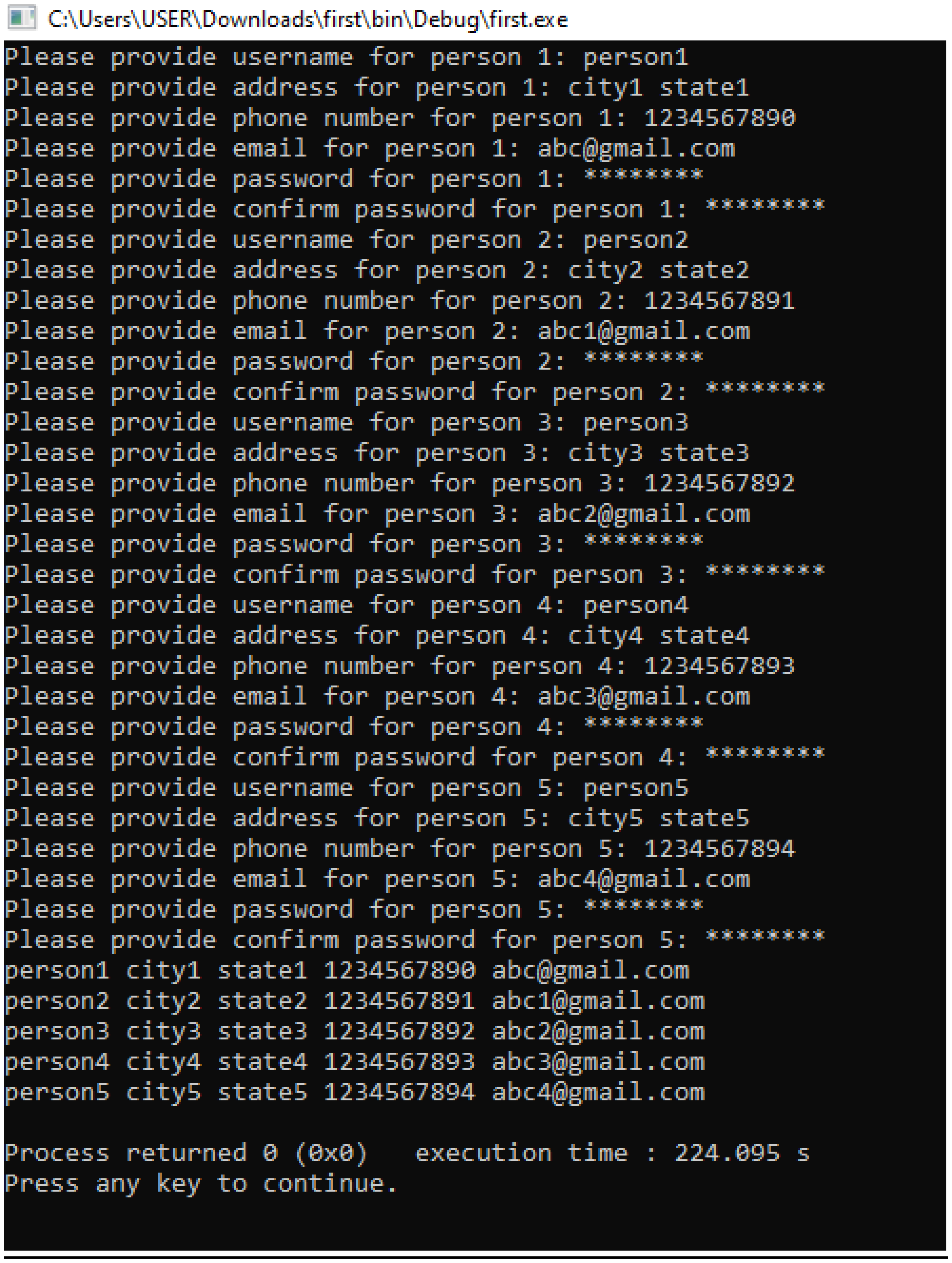
```
#include <stdio.h>
#include <string.h>
char password[9];
struct info {
   char username[100];
   char address[200];
   int phonenumber;
   char email[100];
   char password[9];
   char confirmPass[9];
};
void takePassword() {
   char ch;
   int i = 0;
   int check = 0;
```

```
    while(1) {
        ch = getch();
        if(ch == 13) {
            break;
        } else if(ch == 8) {
            password[--i] = '\0';
            printf("\b \b");
        }else if(ch == 32 || ch == 9) {
            continue;
        }else {

            if((ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9') || (ch >= 'A' && ch <= 'Z')) {
                password[i++] = ch;
                printf("*");
            }else {
                printf("Invalid format");
                break;
            }
        }
    }

    if(i != 8) {
        printf("Invalid password format!\n");
        printf("Please Enter again: ");
        takePassword();
    }else {
        password[i++] = '\0';
    }
}
int main() {
```

```
int users = 5;

struct info arr[5];


for(int i = 0; i < 5; i++) {

   printf("Please provide username for person %d: ", i + 1);

   scanf("%[^\n]%*c", arr[i].username);


   printf("Please provide address for person %d: ", i + 1);

   scanf("%[^\n]%*c", arr[i].address);


   printf("Please provide phone number for person %d: ", i + 1);

   scanf("%d", &arr[i].phonenumber);


   printf("Please provide email for person %d: ", i + 1);

   scanf(" %[^\n]%*c", arr[i].email);


   printf("Please provide password for person %d: ", i + 1);

   takePassword();

   printf("\n");

   strcpy(arr[i].password, password);


   printf("Please provide confirm password for person %d: ", i + 1);

   takePassword();

   printf("\n");


   if(strcmp(password, arr[i].password) != 0) {

      printf("The confirm password is incorrect, please enter info again!!");

      i--;

   }else {

      strcpy(arr[i].confirmPass, password);
```

```
        }
    }
    for(int i = 0; i < 5; i++) {
        printf("%s %s %d %s\n", arr[i].username, arr[i].address, arr[i].phonenumber, arr[i].email);
    }
}
```

Output:



```
C:\Users\USER\Downloads\first\bin\Debug\first.exe
Please provide username for person 1: person1
Please provide address for person 1: city1 state1
Please provide phone number for person 1: 1234567890
Please provide email for person 1: abc@gmail.com
Please provide password for person 1: ********
Please provide confirm password for person 1: ********
Please provide username for person 2: person2
Please provide address for person 2: city2 state2
Please provide phone number for person 2: 1234567891
Please provide email for person 2: abc1@gmail.com
Please provide password for person 2: ********
Please provide confirm password for person 2: ********
Please provide username for person 3: person3
Please provide address for person 3: city3 state3
Please provide phone number for person 3: 1234567892
Please provide email for person 3: abc2@gmail.com
Please provide password for person 3: ********
Please provide confirm password for person 3: ********
Please provide username for person 4: person4
Please provide address for person 4: city4 state4
Please provide phone number for person 4: 1234567893
Please provide email for person 4: abc3@gmail.com
Please provide password for person 4: ********
Please provide confirm password for person 4: ********
Please provide username for person 5: person5
Please provide address for person 5: city5 state5
Please provide phone number for person 5: 1234567894
Please provide email for person 5: abc4@gmail.com
Please provide password for person 5: ********
Please provide confirm password for person 5: ********
person1 city1 state1 1234567890 abc@gmail.com
person2 city2 state2 1234567891 abc1@gmail.com
person3 city3 state3 1234567892 abc2@gmail.com
person4 city4 state4 1234567893 abc3@gmail.com
person5 city5 state5 1234567894 abc4@gmail.com

Process returned 0 (0x0)   execution time : 224.095 s
Press any key to continue.
```

(2) Take the details of 5 students such as name, regno, school, branch, blood group, address and phone number. Ensure that the register numbers are unique. Sort the student records based on register number. (Reg.no should be a string). (Use Structures)

Pseudo code:

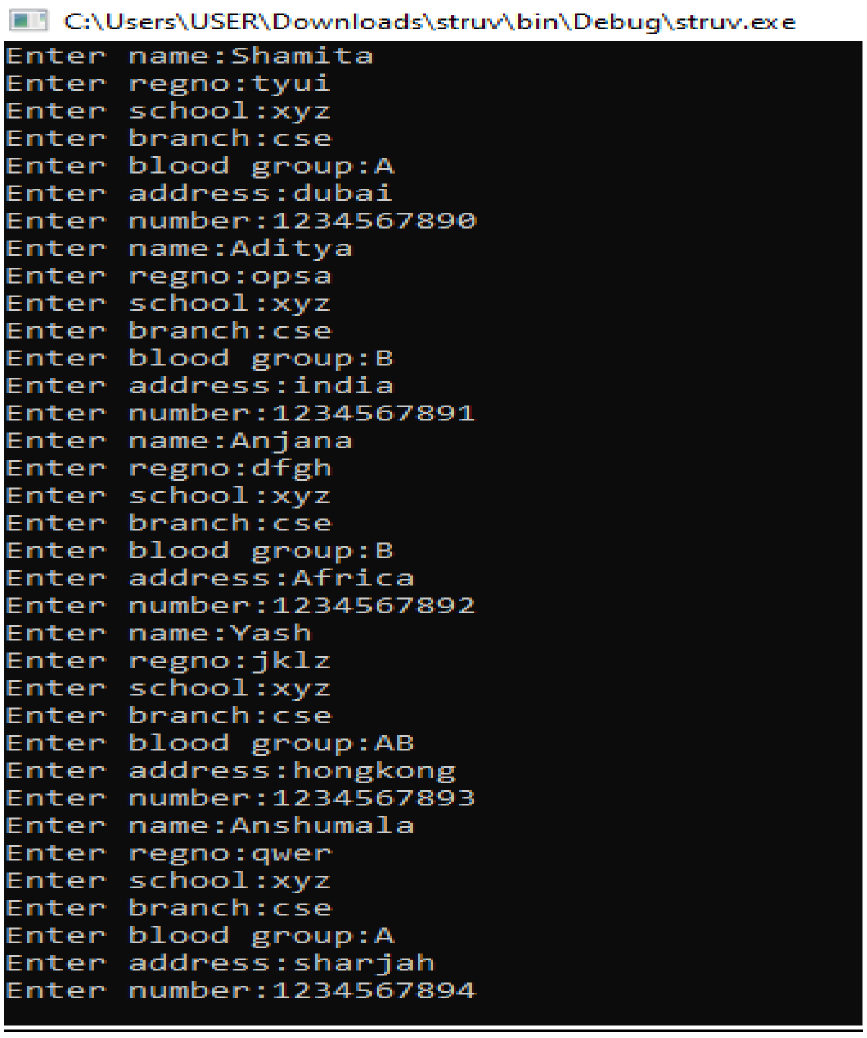
```
1.Begin
2.Initialize structure student
3.Copy the string for the above details to temp
4.Print the details above by taking input from the user
  For registration number
  for(j=0;j<i;j++)
        if(strcmp(s[i].reg,s[j].reg)==0)
                print please enter unique regno
                print Enter regno
                scanf("%s",s[i].reg)
        endif
  endfor
5. for sorting the registration numbers use sortReg(s)
6.Print the sorted list of students based on their registration number
7.End
```

Code:

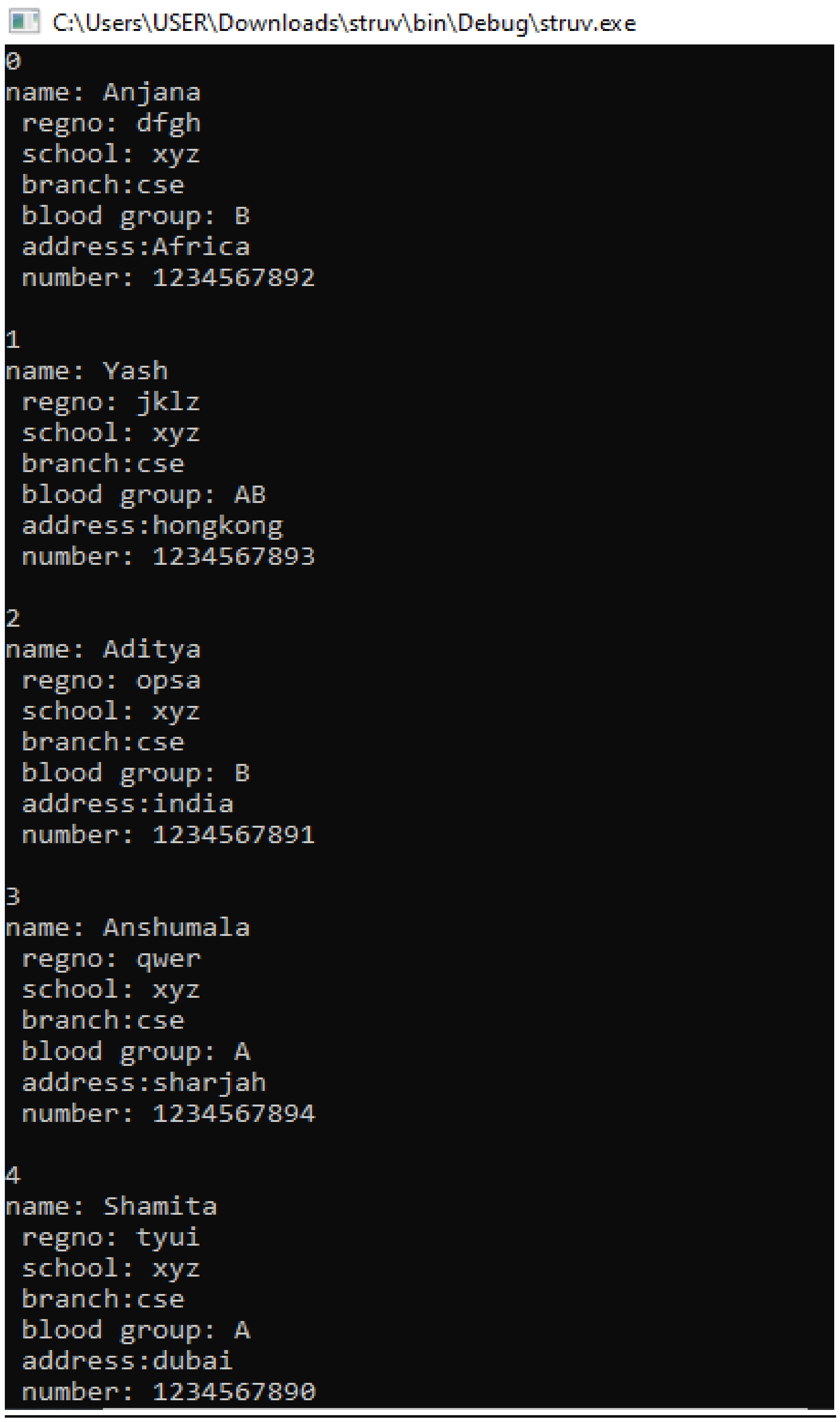
```
#include <stdio.h>
#include <string.h>
struct student{
   char name[100],reg[100],school[50],branch[100],blood[10],address[100],num[100];
}s[5];
void sortReg(struct student s1[]){
  char temp[100];
  for (int i = 0; i < 5; ++i) {
    for (int j = i + 1; j < 5; ++j) {
      if (strcmp(s1[i].reg, s1[j].reg) > 0) {
        strcpy(temp, s1[i].name);
        strcpy(s1[i].name, s1[j].name);
```

```
            strcpy(s1[j].name, temp);

            strcpy(temp, s1[i].reg);

            strcpy(s1[i].reg, s1[j].reg);

            strcpy(s1[j].reg, temp);

            strcpy(temp, s1[i].branch);

            strcpy(s1[i].branch, s1[j].branch);

            strcpy(s1[j].branch, temp);

            strcpy(temp, s1[i].school);

            strcpy(s1[i].school, s1[j].school);

            strcpy(s1[j].school, temp);

            strcpy(temp, s1[i].blood);

            strcpy(s1[i].blood, s1[j].blood);

            strcpy(s1[j].blood, temp);

            strcpy(temp, s1[i].address);

            strcpy(s1[i].address, s1[j].address);

            strcpy(s1[j].address, temp);

            strcpy(temp, s1[i].num);

            strcpy(s1[i].num, s1[j].num);

            strcpy(s1[j].num, temp);


         }

       }

    }

}

int main(){

    int i,j;

    int flag=0;

    for(i=0;i<5;i++){

        printf("Enter name:");

        scanf("%s",s[i].name);
        printf("Enter regno:");
        scanf("%s",s[i].reg);
        for(j=0;j<i;j++){
            if(strcmp(s[i].reg,s[j].reg)==0){
                printf("\nplease enter unique regno\n");
                printf("Enter regno:");
                scanf("%s",s[i].reg);
            }
        }
        printf("Enter school:");
        scanf("%s",s[i].school);
        printf("Enter branch:");
        scanf("%s",s[i].branch);
        printf("Enter blood group:");
        scanf("%s",s[i].blood);
        printf("Enter address:");
        scanf("%s",s[i].address);
        printf("Enter number:");
        scanf("%s",s[i].num);
    }
    sortReg(s);
    for(i=0;i<5;i++){
        printf("\n\n%d\nname: %s\n regno: %s\n school: %s\n branch:%s\n blood group: %s\n address:%s \n
number: %s",i,s[i].name,s[i].reg,s[i].school,s[i].branch,s[i].blood,s[i].address,s[i].num);
    }
    return 0;
}
```

Output:



```
C:\Users\USER\Downloads\struv\bin\Debug\struv.exe
Enter name:Shamita
Enter regno:tyui
Enter school:xyz
Enter branch:cse
Enter blood group:A
Enter address:dubai
Enter number:1234567890
Enter name:Aditya
Enter regno:opsa
Enter school:xyz
Enter branch:cse
Enter blood group:B
Enter address:india
Enter number:1234567891
Enter name:Anjana
Enter regno:dfgh
Enter school:xyz
Enter branch:cse
Enter blood group:B
Enter address:Africa
Enter number:1234567892
Enter name:Yash
Enter regno:jklz
Enter school:xyz
Enter branch:cse
Enter blood group:AB
Enter address:hongkong
Enter number:1234567893
Enter name:Anshumala
Enter regno:qwer
Enter school:xyz
Enter branch:cse
Enter blood group:A
Enter address:sharjah
Enter number:1234567894
```



```
C:\Users\USER\Downloads\struv\bin\Debug\struv.exe
0
name: Anjana
 regno: dfgh
 school: xyz
 branch:cse
 blood group: B
 address:Africa
 number: 1234567892

1
name: Yash
 regno: jklz
 school: xyz
 branch:cse
 blood group: AB
 address:hongkong
 number: 1234567893

2
name: Aditya
 regno: opsa
 school: xyz
 branch:cse
 blood group: B
 address:india
 number: 1234567891

3
name: Anshumala
 regno: qwer
 school: xyz
 branch:cse
 blood group: A
 address:sharjah
 number: 1234567894

4
name: Shamita
 regno: tyui
 school: xyz
 branch:cse
 blood group: A
 address:dubai
 number: 1234567890
```

(3) Write a program in C to reverse the given array.

<u>Pseudo code:</u>

```
1.Begin procedure reverse

2.Read the array

3.Initialize first as 0 and last as n-1

4.while( first<last)
```

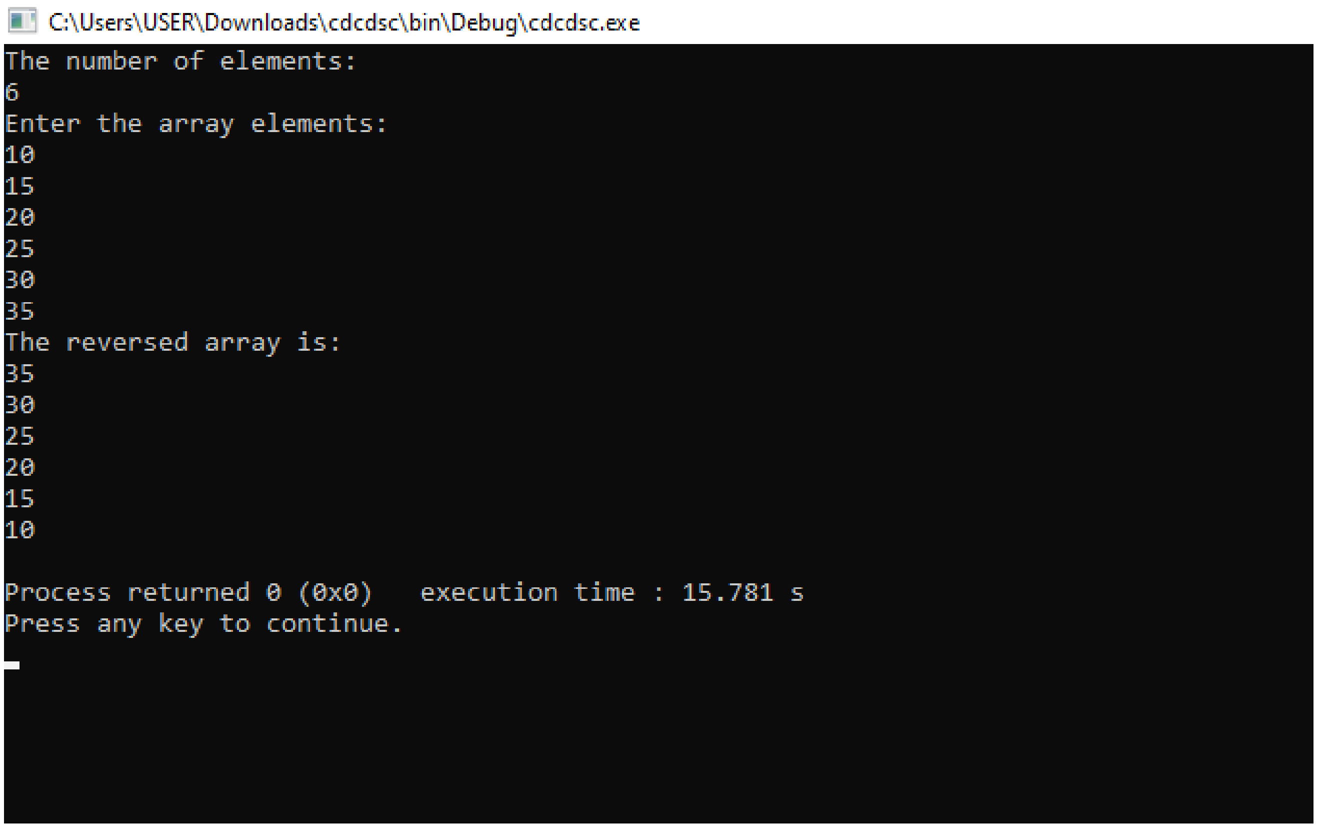
```
        temp=a[first]
        a[first]=a[end]
        a[end]=temp
        first increment
        last decrement
end while
5.Print the reversed array
6.End procedure
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int a[10],first,last,n,temp;
printf("The number of elements:\n");
scanf("%d",&n);

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

first=0;
last=n-1;
  while(first<last)
  {
      temp=a[first];
      a[first]=a[last];
      a[last]=temp;
      first++;
      last--;
```

```
 }
    printf("The reversed array is:\n");
    for(first=0;first<n;first++)
    printf("%d\n",a[first]);
 return 0;
}
```

Output:



C:\Users\USER\Downloads\cdcdsc\bin\Debug\cdcdsc.exe

```
The number of elements:
6
Enter the array elements:
10
15
20
25
30
35
The reversed array is:
35
30
25
20
15
10

Process returned 0 (0x0)   execution time : 15.781 s
Press any key to continue.
```

(4) Write a program to rotate (arr[], d, n) that rotates arr[] of size n by d elements.(Shifting each elements by 2 location)

Pseudo code:

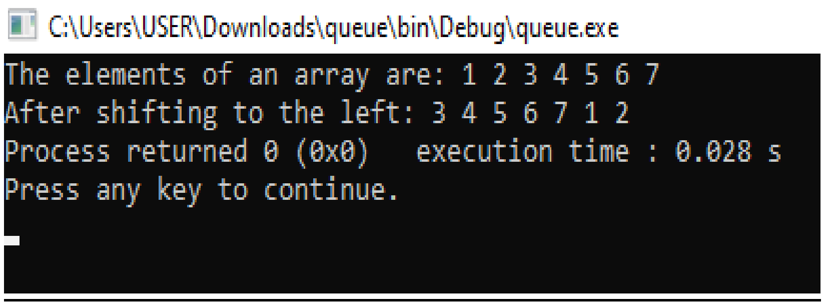
```
1.Begin procedure left rotation.
2.Read the array
3.For(j=1;j<=2;j++)
      temp=a[0]
      for(i=0;i<6;i++)
            a[i]= a[i+1]
      end for
```

```
        a[i]= temp
end for
4.Print the elements after shifting
5.End
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int a[7]={1,2,3,4,5,6,7};
   int i=0,j=0,temp;
   printf("The elements of an array are: ");
   for(i=0;i<7;i++)
      printf("%d ",a[i]);
      printf("\n");
   for(j=1;j<=2;j++)
   {temp=a[0];
   for(i=0;i<6;i++)
   {
      a[i]=a[i+1];
   }
   a[i]=temp;
   }
   printf("After shifting to the left: ");
   for(i=0;i<7;i++)
      printf("%d ",a[i]);
return 0;
}
```

Output:



```
The elements of an array are: 1 2 3 4 5 6 7
After shifting to the left: 3 4 5 6 7 1 2
Process returned 0 (0x0)   execution time : 0.028 s
Press any key to continue.
```

(5) Write an Algorithm to Split the array and add the first part to the end.

Pseudo code:

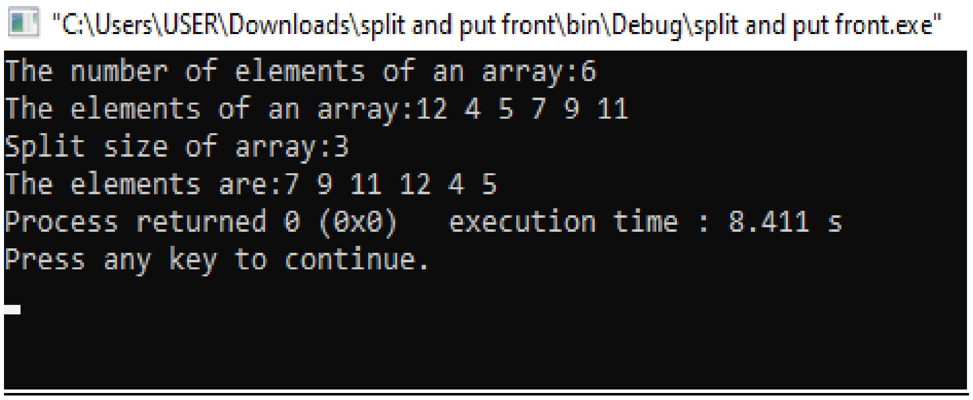
```
1.Begin
2.Read the array
3. Check if (r>n)
        print Split size greater than array size
   endif
4. Check(r<n)
        for(i=r;i<n;i++)
            a2[i-r]= a1[i]
        end for
        for(i=n-r;i<n;i++)
            a2[i]= a1[i-(n-r)]
        end for
5.Print the updated array
6.End
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int n,i,r;
  printf("The number of elements of an array:");
  scanf("%d",&n);
  int a1[n],a2[n];
```

```
    printf("The elements of an array:");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a1[i]);
    }
    printf("Split size of array:");
    scanf("%d",&r);
    if(r>n)
    {
        printf("Split size greater than array size");
    }
    else
    {
        for(i=r;i<n;i++)
        {
            a2[i-r]=a1[i];
        }
        for(i=n-r;i<n;i++)
        {
            a2[i]=a1[i-(n-r)];
        }
        printf("The elements are:");
        for(i=0;i<n;i++)
        {
          printf("%d ",a2[i]);
        }
    }
}
```

Output:



"C:\Users\USER\Downloads\split and put front\bin\Debug\split and put front.exe"

```
The number of elements of an array:6
The elements of an array:12 4 5 7 9 11
Split size of array:3
The elements are:7 9 11 12 4 5
Process returned 0 (0x0)   execution time : 8.411 s
Press any key to continue.
```

(6) Write a C program to print all the repeated numbers with frequency in an array.

Pseudo code:

```
1.Begin
2. Read the elements in the array and initialize frequency and initialize frequency as -1
3.  for(i=0; i<size; i++)
        count = 1
        for(j=i+1; j<size; j++)
            if(arr[i]==arr[j])
                count++
                freq[j] = 0
            endif
        end for
        if(freq[i] != 0)
            freq[i] = count
        endif
    end for
4. print the frequency of elements of array
5. end
```

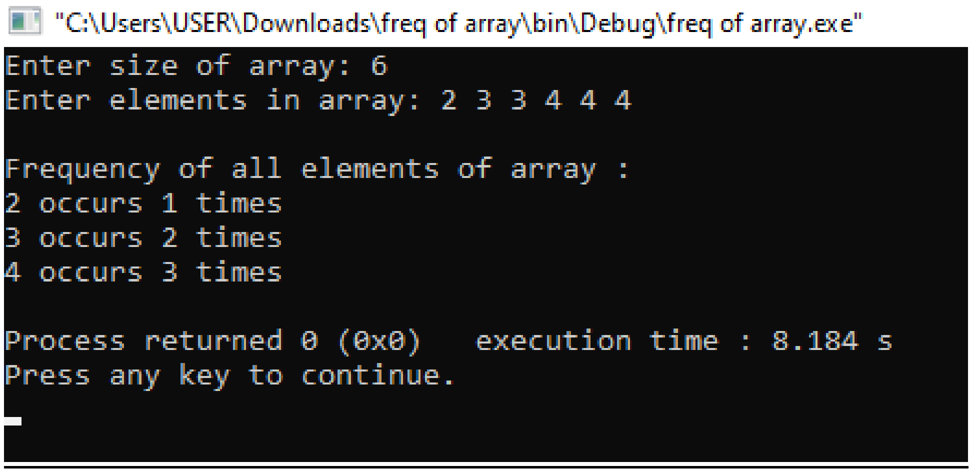
Code:

```
#include <stdio.h>
int main()
{
```

```
int arr[100], freq[100];
int size, i, j, count;
printf("Enter size of array: ");
scanf("%d", &size);
printf("Enter elements in array: ");
for(i=0; i<size; i++)
{
    scanf("%d", &arr[i]);
    freq[i] = -1;
}
for(i=0; i<size; i++)
{
    count = 1;
    for(j=i+1; j<size; j++)
    {
        if(arr[i]==arr[j])
        {
            count++;
            freq[j] = 0;
        }
    }
    if(freq[i] != 0)
    {
        freq[i] = count;
    }
}
printf("\nFrequency of all elements of array : \n");
for(i=0; i<size; i++)
{
    if(freq[i] != 0)
```

```
      {
         printf("%d occurs %d times\n", arr[i], freq[i]);
      }
   }

   return 0;
}
```

Output:



"C:\Users\USER\Downloads\freq of array\bin\Debug\freq of array.exe"

```
Enter size of array: 6
Enter elements in array: 2 3 3 4 4 4

Frequency of all elements of array :
2 occurs 1 times
3 occurs 2 times
4 occurs 3 times

Process returned 0 (0x0)   execution time : 8.184 s
Press any key to continue.
```

(7) Write a C program to implement

(1). Insertion of an Element into the array (with position)

(2). Delete an element from the array (With position)

(3). Find or Search an Element in the given array.

i) Binary search

ii) Linear search

(4). Update the k th element in the given array (given position)

(5). Display the next and previous element based on the position given.

Pseudo code:

```
1.Begin
2. while(1)
        Print statements for insert,delete,linear,binary,update,display element beside a given position and exit
        Initiate switch cases to choose a particular choice
    End while
3.procedure insert()
   Read the elements of the array
   for (i=9;i>=c;i--)
          a[i] = a[i-1]
   endfor
   a[c-1] = b
  print the updated array
4.procedure delete()
    Read the array
     for (i=e-1;i<9;i++)
          a[i] = a[i+1]
      endfor
      print the updated array
5.procedure linearsearch()
    Read the array
       while(1)
           if (a[count]==b)
                    print (Yes %d is present in the array,b)
                    printf(The position of the number is %d,count+1)
          break
          endif
       count +=1
       if (count > 10)
           Print The number is not in the given Array
```

```
            Break
        Endif
      Endwhile
6.procedure binarysearch()
    Read the array
    while (count < 10)
            count +=1
            mid = (high + low)/2
            if (a[mid]>b)
                    high = mid
                    low = 0
            endif
        if (a[mid] < b)
            high = 9
            low = mid;
        endif
        if (a[mid]==b\)
            print Yes the number belongs in the Array at position %d,mid+1);
            count = 10
             x = 0
        endif
        endwhile
 if (x==0)
endif
otherwise print The number %d is not present in the array
7.procedure update()
```

```
Input the array
Input the position and the element to be updated
a[pos-1]=ele
print the updated array
8.procedure elementprint()
  Read the array
  Enter the position
   Print a[b]
   Print a[b-2]
9.End
```

Code:

```
#include<stdio.h>
int main()
{
    while(1)
    {
        int choice;
        printf("Please select a number between 1-7\n");
        printf("1.Insert a number\n");
        printf("2.Delete a number\n");
        printf("3.Linear Search\n");
        printf("4.Binary Search\n");
        printf("5.Update\n");
        printf("6.Display elements beside a given position\n");
        printf("7.Exit\n");
        scanf("%d",&choice);
        switch(choice)

        {

        case 1:

           insert();

           break;

        case 2:

           Delete();

           break;

        case 3:

           linearsearch();

           break;

        case 4:

           Binarysearch();

           break;

        case 5:

           update();

           break;

        case 6:

           elementprint();

           break;

        case 7:

           exit(1);

        }

     }

  }
```

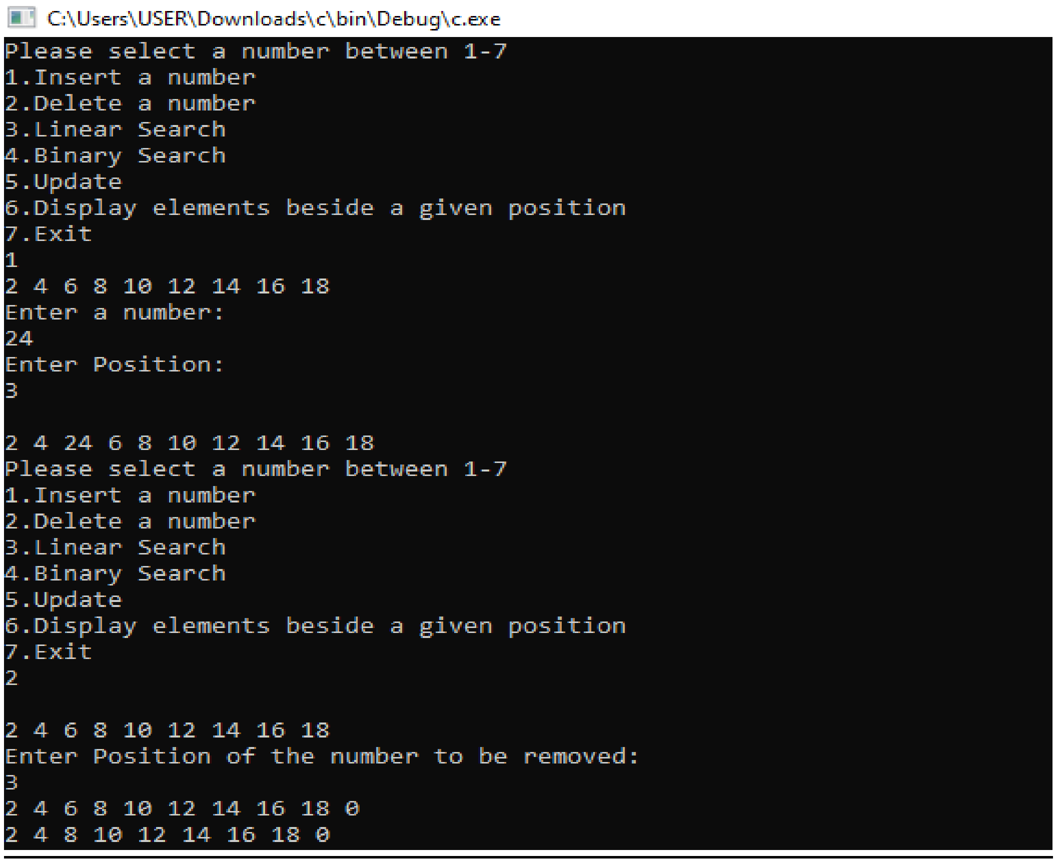
```
void insert()

{

   int a[10]={2,4,6,8,10,12,14,16,18},b,c,count=0,d[10],i;

   for (i = 0;i<9;i++)

   {

      printf("%d",a[i]);

      printf(" ");

   }

   printf("\nEnter a number:\n");

   scanf("%d",&b);

   printf("Enter Position:\n");

   scanf("%d",&c);

   printf("\n");

   for (i=9;i>=c;i--)

   {

      a[i] = a[i-1];

   }


   a[c-1] = b;

   for(i=0;i<10;i++)

   {

      printf("%d",a[i]);

      printf(" ");

   }

   printf("\n");

}
```

```
void Delete()

{

    int a[10]={2,4,6,8,10,12,14,16,18};

    printf("\n");

    int e,i;

    for (i = 0;i<9;i++)

    {

        printf("%d",a[i]);

        printf(" ");

    }

    printf("\n");

    printf("Enter Position of the number to be removed:\n");

    scanf("%d",&e);

    for(i=0;i<10;i++)

    {

        printf("%d",a[i]);

        printf(" ");

    }

    printf("\n");

    for (i=e-1;i<9;i++)

    {

        a[i] = a[i+1];

    }

    for(i=0;i<9;i++)

    {

        printf("%d",a[i]);
        printf(" ");

    }

    printf("\n");


return 0;


}

void linearsearch()

{

    int a[10] ={2,21,78,13,35,80,79,86,98,111},i;

    int b,count=0;

    for (i = 0;i<9;i++)

    {

        printf("%d",a[i]);

        printf(" ");

    }

    printf("\nEnter a number:\n");

    scanf("%d",&b);

    while(1)

    {

        if (a[count]==b)

        {

            printf("\nYes %d is present in the array\n",b);

            printf("The position of the number is %d\n",count+1);

            break;

        }
        count +=1;
        if (count > 10)
        {
            printf("\nThe number is not in the given Array");
            break;
        }
    }
}
void Binarysearch()
{
    int a[10] = {2,21,78,13,35,80,79,86,98,111},i;
    int b,count=0,low=0,high=9,mid,x = 1;
    for (i = 0;i<9;i++)
    {
        printf("%d",a[i]);
        printf(" ");
    }
    printf("\nEnter a number:\n");
    scanf("%d",&b);
    while (count < 10)
    {
        count +=1;
        mid = (high + low)/2;
        printf("\n");
        if (a[mid]>b)
        {
            high = mid;

            low = 0;

        }

        if (a[mid] < b)

        {

            high = 9;

            low = mid;

        }

        if (a[mid]==b\

            )

        {

            printf("\nYes the number belongs in the Array at position %d\n",mid+1);

            count = 10;

            x = 0;

        }

    }

    if (x==0)

    {


    }

    else

    {

        printf("\nThe number %d is not present in the array",b);

    }

}

void update()
```

```
{
    int n,i,a[10],pos,ele;
    printf("Enter the size of array:");
    scanf("%d",&n);
    printf("Enter the elements of array:");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("Enter the position:");
    scanf("%d",&pos);
    printf("Enter the element:");
    scanf("%d",&ele);
    a[pos-1]=ele;
    printf("After updating array:");
    for(i=0;i<n;i++)
        printf(" %d\n",a[i]);
}
void elementprint()
{
    int a[10] = {1,2,3,4,5,6,7,8,9};
    int b,c;
    for (int i=0;i<9;i++)
    {
        printf("%d",a[i]);
        printf(" ");
    }
    printf("\n");
```

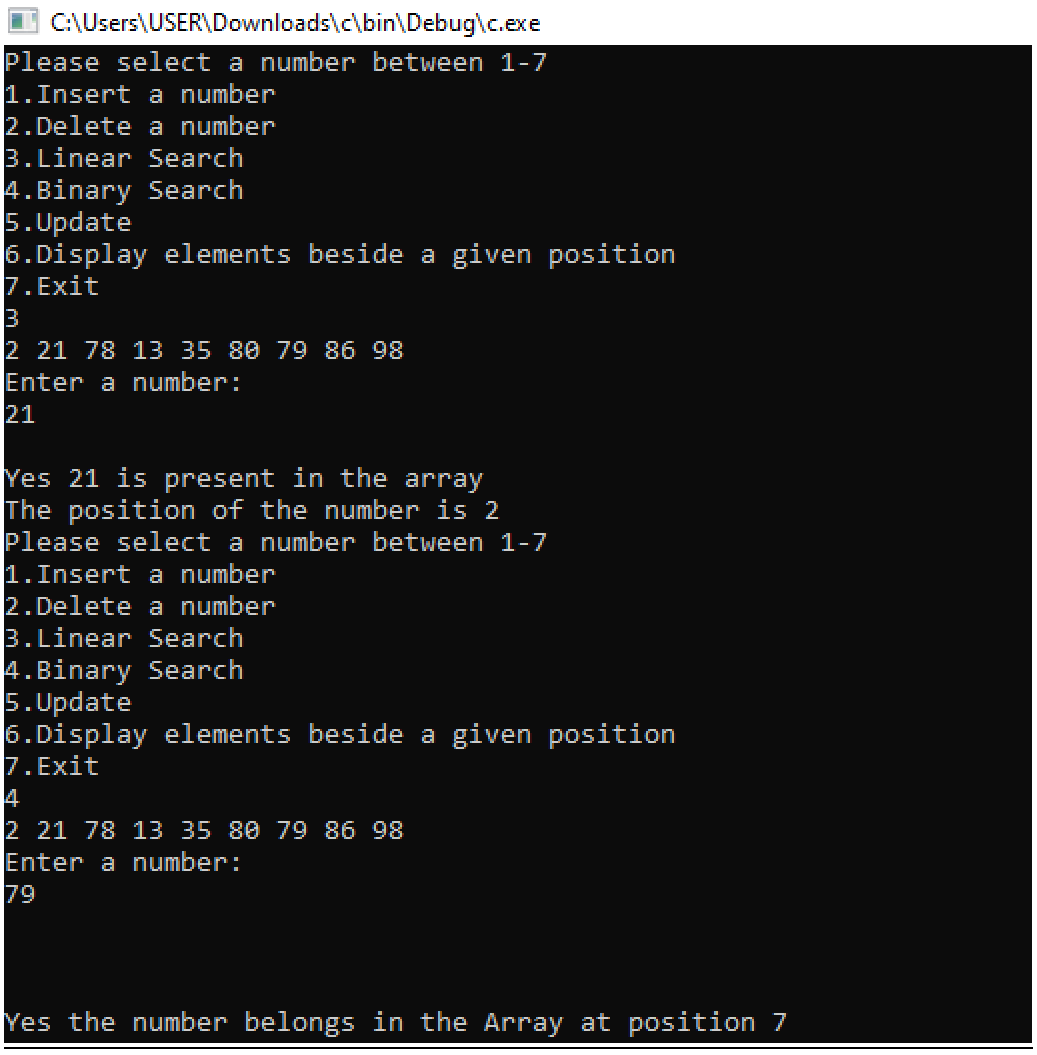
```
   printf("Enter the position of the element:\n");

   scanf("%d",&b);

   printf("%d",a[b]);

   printf("\n");

   printf("%d",a[b-2]);

}
```

Output:



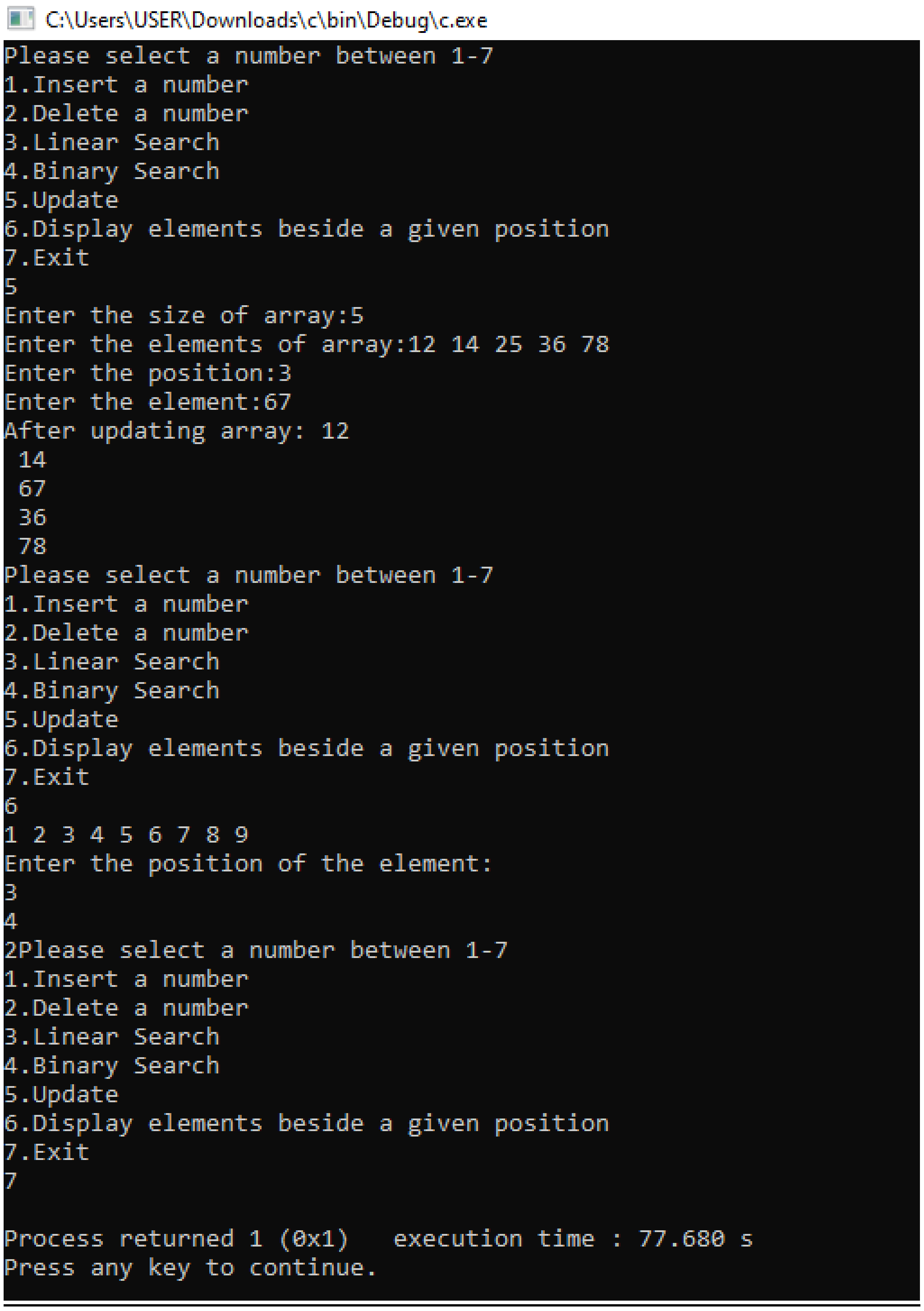
C:\Users\USER\Downloads\c\bin\Debug\c.exe

```
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
1
2 4 6 8 10 12 14 16 18
Enter a number:
24
Enter Position:
3

2 4 24 6 8 10 12 14 16 18
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
2

2 4 6 8 10 12 14 16 18
Enter Position of the number to be removed:
3
2 4 6 8 10 12 14 16 18 0
2 4 8 10 12 14 16 18 0
```



C:\Users\USER\Downloads\c\bin\Debug\c.exe

```
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
3
2 21 78 13 35 80 79 86 98
Enter a number:
21

Yes 21 is present in the array
The position of the number is 2
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
4
2 21 78 13 35 80 79 86 98
Enter a number:
79



Yes the number belongs in the Array at position 7
```



```
C:\Users\USER\Downloads\c\bin\Debug\c.exe
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
5
Enter the size of array:5
Enter the elements of array:12 14 25 36 78
Enter the position:3
Enter the element:67
After updating array: 12
 14
 67
 36
 78
Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
6
1 2 3 4 5 6 7 8 9
Enter the position of the element:
3
4
2Please select a number between 1-7
1.Insert a number
2.Delete a number
3.Linear Search
4.Binary Search
5.Update
6.Display elements beside a given position
7.Exit
7

Process returned 1 (0x1)   execution time : 77.680 s
Press any key to continue.
```

(8) Write C program to implement stack using an array with the operation:

i) Push

ii) Pop

iii) isEmpty()

iv) isFull()

Pseudo code:

```
1.Begin
2.Initialize stack structure.
3. procedure isEmpty(struct stack* ptr)
   if(ptr->top == -1)
         return 1
   endif
   end procedure
4. procedure isFull(struct stack* ptr)
   if(ptr->top == ptr->size - 1)
      return 1
  endif
  end procedure
5. procedure push(struct stack* ptr, int val){
   if(isFull(ptr))
      print Stack Overflow! Cannot push %d to the stack\n", val
  endif
   else
      ptr->top++;
      ptr->arr[ptr->top] = val
   end procedure
6. procedure pop(struct stack* ptr)
   if(isEmpty(ptr))
      print Stack Underflow! Cannot pop from the stack
```

```
        return -1
    endif

    else
        int val = ptr->arr[ptr->top]
        ptr->top--
        return val
 end procedure
```

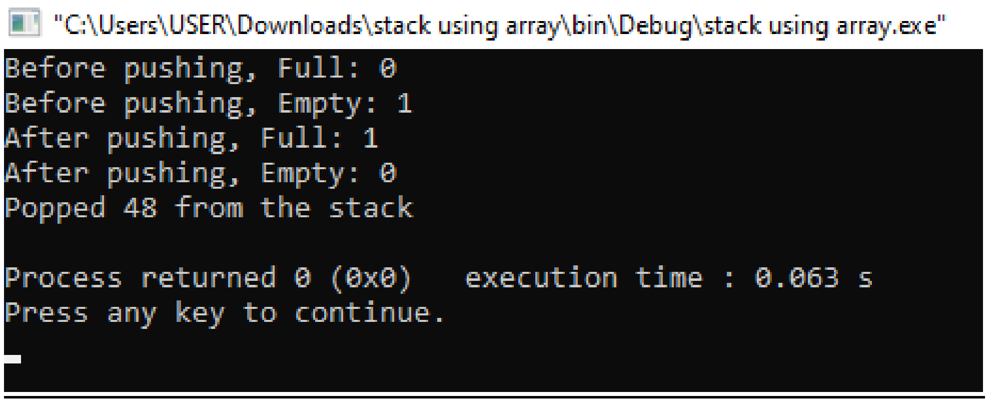
7.Initialize the procedures, variables and elements to be pushed.

8.End

Code:

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

struct stack{
    int size ;
    int top;
    int * arr;
};
int isEmpty(struct stack* ptr){
    if(ptr->top == -1){
            return 1;
        }
        else{
            return 0;
        }
}
int isFull(struct stack* ptr){
    if(ptr->top == ptr->size - 1){
        return 1;
    }
    else{
        return 0;
    }
}
void push(struct stack* ptr, int val){
    if(isFull(ptr)){
        printf("Stack Overflow! Cannot push %d to the stack\n", val);
    }
    else{
        ptr->top++;
        ptr->arr[ptr->top] = val;
    }
}
int pop(struct stack* ptr){
    if(isEmpty(ptr)){
        printf("Stack Underflow! Cannot pop from the stack\n");
        return -1;
    }
    else{
        int val = ptr->arr[ptr->top];
        ptr->top--;
        return val;
    }
}
int main(){
    struct stack *sp = (struct stack *) malloc(sizeof(struct stack));
    sp->size = 8;
    sp->top = -1;
    sp->arr = (int *) malloc(sp->size * sizeof(int));
    printf("Before pushing, Full: %d\n", isFull(sp));
    printf("Before pushing, Empty: %d\n", isEmpty(sp));
    push(sp, 10);
    push(sp, 3);
    push(sp, 9);
    push(sp, 77);
    push(sp, 30);
    push(sp, 4);
    push(sp, 52);
    push(sp, 48);
    printf("After pushing, Full: %d\n", isFull(sp));
    printf("After pushing, Empty: %d\n", isEmpty(sp));
    printf("Popped %d from the stack\n", pop(sp));
    return 0;
}
```

Output:



```
"C:\Users\USER\Downloads\stack using array\bin\Debug\stack using array.exe"
Before pushing, Full: 0
Before pushing, Empty: 1
After pushing, Full: 1
After pushing, Empty: 0
Popped 48 from the stack

Process returned 0 (0x0)   execution time : 0.063 s
Press any key to continue.
```

(9) Write a C Program to implement Queue with the following operation :(Design a Menu driven program)

i) Inserting and Element

ii) Deleting an Element

iii) Display the Element

iv) Exit

<u>Pseudo code:</u>

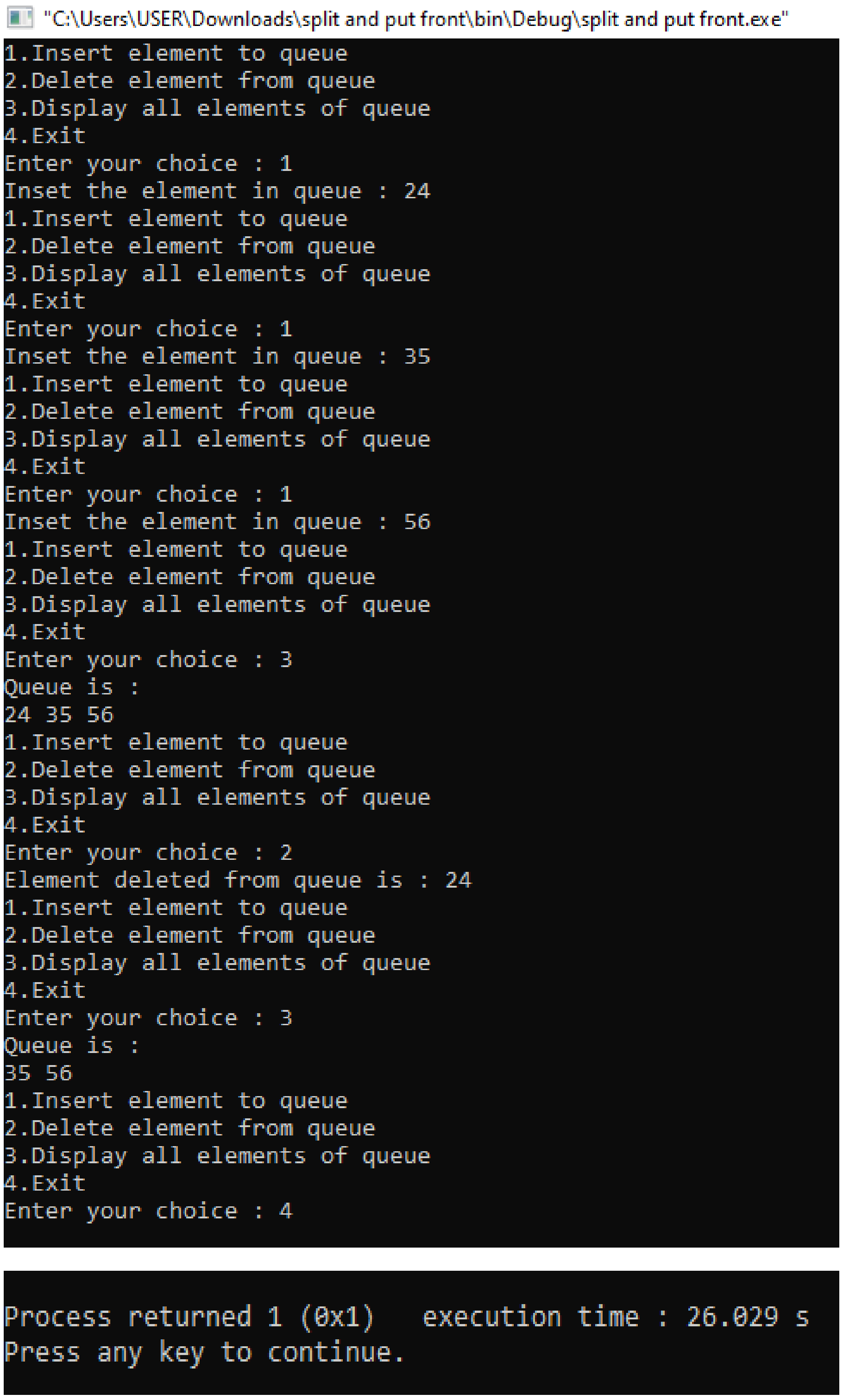
```
1.Begin
2.initialize rear and front as -1
3. while(1)
        Print statements for push,pop,display and exit
        Initiate switch cases to choose a particular choice
  End while
4. procedure enqueue(data)
  if queue is full
    return overflow
  endif
  rear ← rear + 1
  queue[rear] ← data
  return true
end procedure
5. procedure dequeue
  if queue is empty
    return underflow
  end if
  data = queue[front]
  front ← front + 1
  return true
end procedure
```

```
6.procedure display()
  If front==-1
            Print queue is empty
 Endif
Print the elements of queue from front to rear
7.Exit
```

Code:

```
#include <stdio.h>
#define MAX 10
int queue_array[MAX];
            int rear = - 1;
            int front = - 1;
main()
{
    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Exit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
switch (choice)
        {
            case 1:
                        insert();
                        break;
            case 2:
                        delete();
                        break;
            case 3:
                        display();
                        break;
            case 4:
                         exit(1);
            default:
                        printf("Wrong choice \n");
        }
    }
}
void insert()
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
}
void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
}
void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
}
```

Output:



```
"C:\Users\USER\Downloads\split and put front\bin\Debug\split and put front.exe"
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1
Inset the element in queue : 24
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1
Inset the element in queue : 35
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1
Inset the element in queue : 56
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 3
Queue is :
24 35 56
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 2
Element deleted from queue is : 24
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 3
Queue is :
35 56
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 4

Process returned 1 (0x1)   execution time : 26.029 s
Press any key to continue.
```

(10) Write a C program to implement balancing the parenthesis using Stack.

<u>Pseudo code:</u>

```
1.Begin
2.Initialize the structure stack
3.procedure push(char item)
    if (s.top == (MAX - 1))
        Print Stack is Full
    endif
    Otherwise
    s.top + 1= s.top
    s.stk[s.top]= item
4.procedure pop()
    Check if (s.top == - 1)
        Print Stack is Empty
    endif
    Otherwise s.top = s.top – 1
5. Enter the expression
6. for(i = 0;i < strlen(exp);i++)
    if(exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
        push(exp[i])
        continue
    endif
    else if(exp[i] == ')' || exp[i] == ']' || exp[i] == '}')
        if each of the parenthesis are matching the top
            pop
        endif
        otherwise print unbalanced expression
        if(s.top == -1)
            print balanced expression
        endif
        endelseif
end for
7.end
```

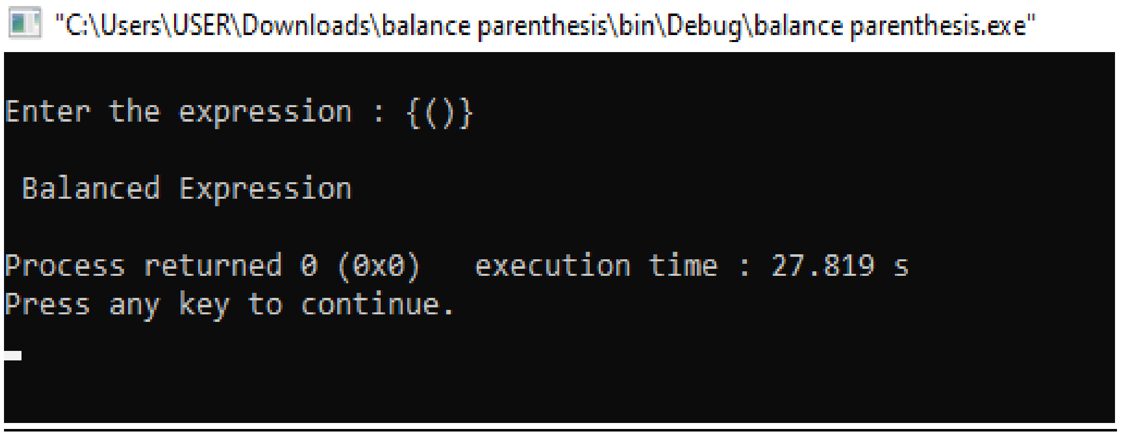
Code:

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 20


struct stack

{

char stk[MAX];

int top;

}s;


void push(char item)

{

if (s.top == (MAX - 1))

printf ("Stack is Full\n");

else

{

s.top = s.top + 1;

s.stk[s.top] = item;

}}


void pop()

{

if (s.top == - 1)

{

printf ("Stack is Empty\n");
```

```
}
else
{
s.top = s.top - 1;
}}

int main()
{
char exp[MAX];
int i = 0;
s.top = -1;
printf("\nEnter the expression : ");
scanf("%s", exp);
for(i = 0;i < strlen(exp);i++)
{
if(exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
{
push(exp[i]);
continue;
}
else if(exp[i] == ')' || exp[i] == ']' || exp[i] == '}')
{
if(exp[i] == ')')
{
if(s.stk[s.top] == '(')
{
pop();
}
else
{
```

```
printf("\nUnbalanced Expression\n");

break;

}}

if(exp[i] == ']')

{

if(s.stk[s.top] == '[')

{

pop();

}

else

{

printf("\nUnbalanced Expression\n");

break;

}}

if(exp[i] == '}')

{

if(s.stk[s.top] == '{')

{

pop();

}

else

{

printf("\n Unbalanced Expression\n");

break;

}}}}

if(s.top == -1)

{

printf("\n Balanced Expression\n");

}}
```

Output:



```
"C:\Users\USER\Downloads\balance parenthesis\bin\Debug\balance parenthesis.exe"

Enter the expression : {()}

 Balanced Expression

Process returned 0 (0x0)   execution time : 27.819 s
Press any key to continue.
```

(11)Write a C Program to implement Infix to postfix conversion of given arithmetic expression.

Pseudo code:

```
1.Begin
2. procedure push(char x)
     stack[++top] =x
3. procedure pop()
     if(top == -1)
          return -1
     endif
     otherwise return stack[top--]
4. procedure priority(char x)
     check if(x == '(')
          return 0
     endif
     if(x == '+' || x == '-')
          return 1
     endif
     if(x == '*' || x == '/')
          return 2
     endif
5. while(*e != '\0')
     if(isalnum(*e))
```

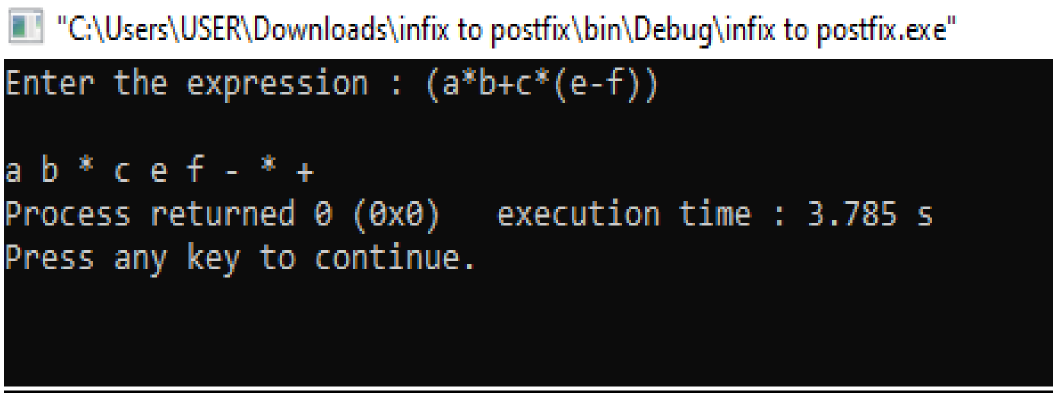
```
                    print("%c ",*e)
          endif
          else if(*e == '(')
                   push(*e)
          end elseif
          else if(*e == ')')
              while((x = pop()) != '(')
              print("%c ", x)
          end elseif
          else
            while(priority(stack[top]) >= priority(*e))
             print("%c ",pop())
             push(*e)
         then increment e
end while
6.   while(top != -1)
          print("%c ",pop());
   end while
7.End
```

Code:

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

char stack[100];
int top = -1;

void push(char x)
{
   stack[++top] = x;
}
```

```
char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
    printf("\n");
    e = exp;

    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c ",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c ",pop());
            push(*e);
        }
        e++;
    }

    while(top != -1)
    {
        printf("%c ",pop());
    }
    return 0;
}
```

Output:



```
"C:\Users\USER\Downloads\infix to postfix\bin\Debug\infix to postfix.exe"
Enter the expression : (a*b+c*(e-f))

a b * c e f - * +
Process returned 0 (0x0)   execution time : 3.785 s
Press any key to continue.
```