INDEX

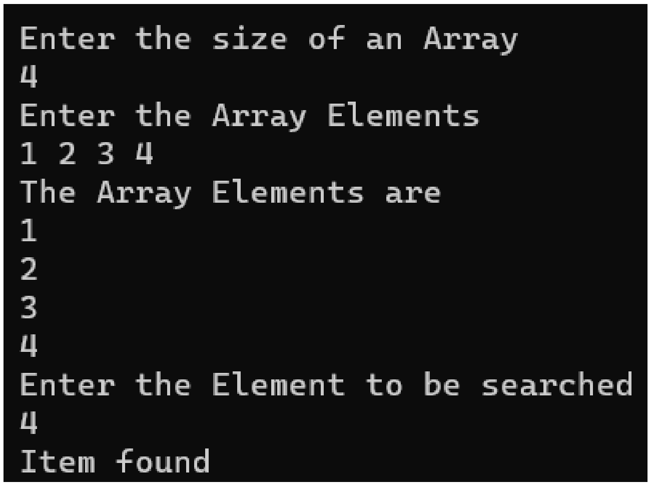
1. Design, Develop and Implement a menu driven Program in C for the following array operations.

A) Creating an array of N Integer Elements
B) Display of array Elements with Suitable Headings
C) Inserting an Element (ELEM) at a given valid Position (POS)
D) D) Deleting an Element at a given valid Position (POS)
E) Exit.

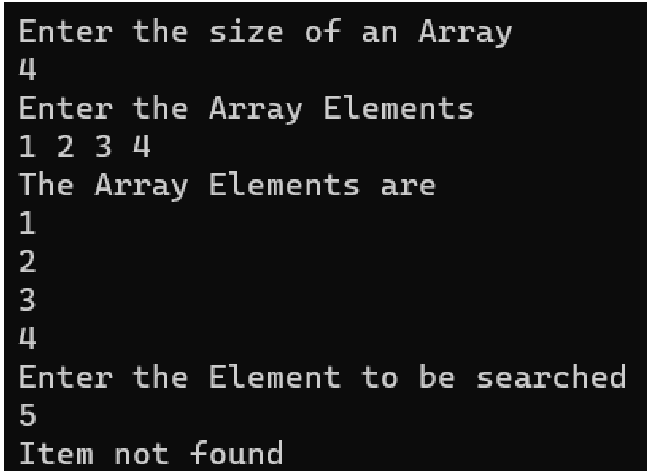
Support the program with functions for each of the above operations

```
#include<stdio.h
#include<stdio.h>
#include<conio.h>
int search( int item, int a[ ], int n)
{
int low, high,key,mid;
low = 0; //Initialization
high = n-1; // Initialization
key=item;
while( low <= high )
{
mid = ( low + high ) / 2; // Find the mid-point
if ( key == a[mid] )
{
 // If item not found, return position
return mid;
}
 if( key < a[mid] )
 high = mid - 1; // Search left side
else
low = mid + 1; // Search right side
}
return -1; // Item not found
}
void main( )
{
 int i,item,a[10],n,pos;
 printf("Enter the size of an Array\n");
 scanf("%d",&n);
 printf("Enter the Array Elements\n");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
 printf("The Array Elements are\n");
 for(i=0;i<n;i++)
 printf("%d\n",a[i]);
 printf("Enter the Element to be searched\n");
 scanf("%d",&item);
 pos=search(item,a,n);
 if(pos==-1)
 printf("Item not found\n");
 else
 printf("Item found\n");
 getch( );
}
```

Output:



```
Enter the size of an Array
4
Enter the Array Elements
1 2 3 4
The Array Elements are
1
2
3
4
Enter the Element to be searched
4
Item found
```



```
Enter the size of an Array
4
Enter the Array Elements
1 2 3 4
The Array Elements are
1
2
3
4
Enter the Element to be searched
5
Item not found
```

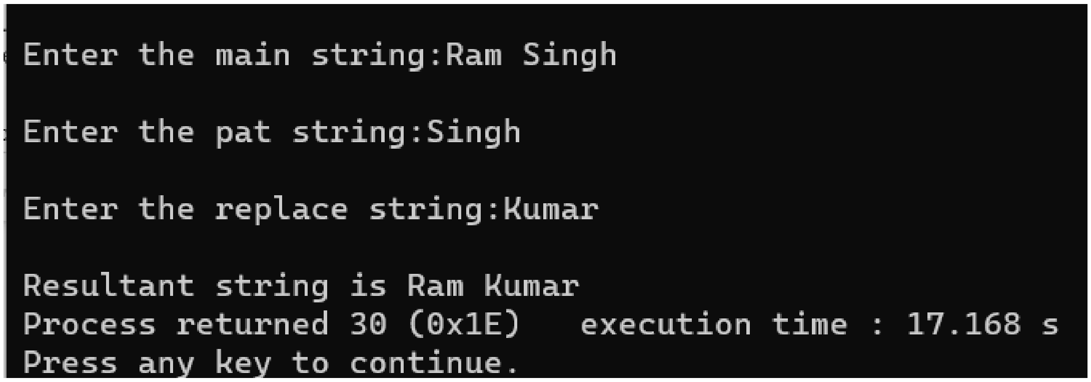
2. Design, develop and Implement a Program in C for the following operations on Strings.

A) Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)

B) Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR.
Report suitable messages in case PAT does not exist in STR
Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
#include<conio.h>
char str[50], pat[20], rep[20], ans[50];
int c=0, m=0, i=0, j=0, k, flag=0;
void stringmatch(){
while(str[c] !='\0'){
if(str[m] == pat[i]){
i++;
m++;
if(pat[i] == '\0'){
flag = 1;
for(k=0; rep[k]!='\0'; k++, j++){
ans[j] = rep[k];
}
i = 0; c = m;
}
}
else{
ans[j]= str[c];
j++;
 c++;
m=c;
i=0;
}
 }
ans[j]='\0';
}
void main(){
printf("\nEnter the main string:"); gets(str);
printf("\nEnter the pat string:");
gets(pat);
printf("\nEnter the replace string:"); gets(rep);
stringmatch();
if(flag == 1)
printf("\nResultant string is %s", ans); else
printf("\nPattern string is not found");
}
```

Output:



```
Enter the main string:Ram Singh

Enter the pat string:Singh

Enter the replace string:Kumar

Resultant string is Ram Kumar
Process returned 30 (0x1E)   execution time : 17.168 s
Press any key to continue.
```

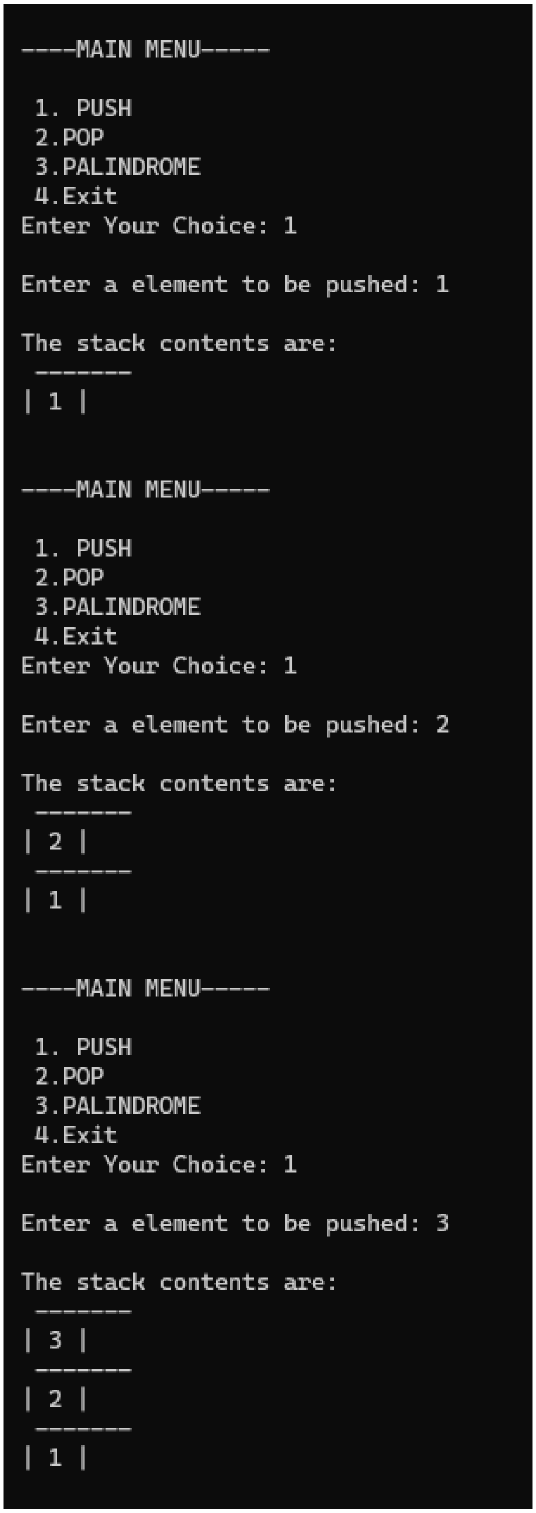
3. Design, Develop and Implement a menu driven Program in C for the following operations on
STACK of Integers (Array Implementation of Stack with maximum size MAX)

- **A) Push an Element on to Stack**
- **B) Pop an Element from Stack**
- **C) Demonstrate how Stack can be used to check Palindrome**
- **D) Demonstrate Overflow and Underflow situations on Stack**
- **E) Display the status of Stack**
- **F) Exit**

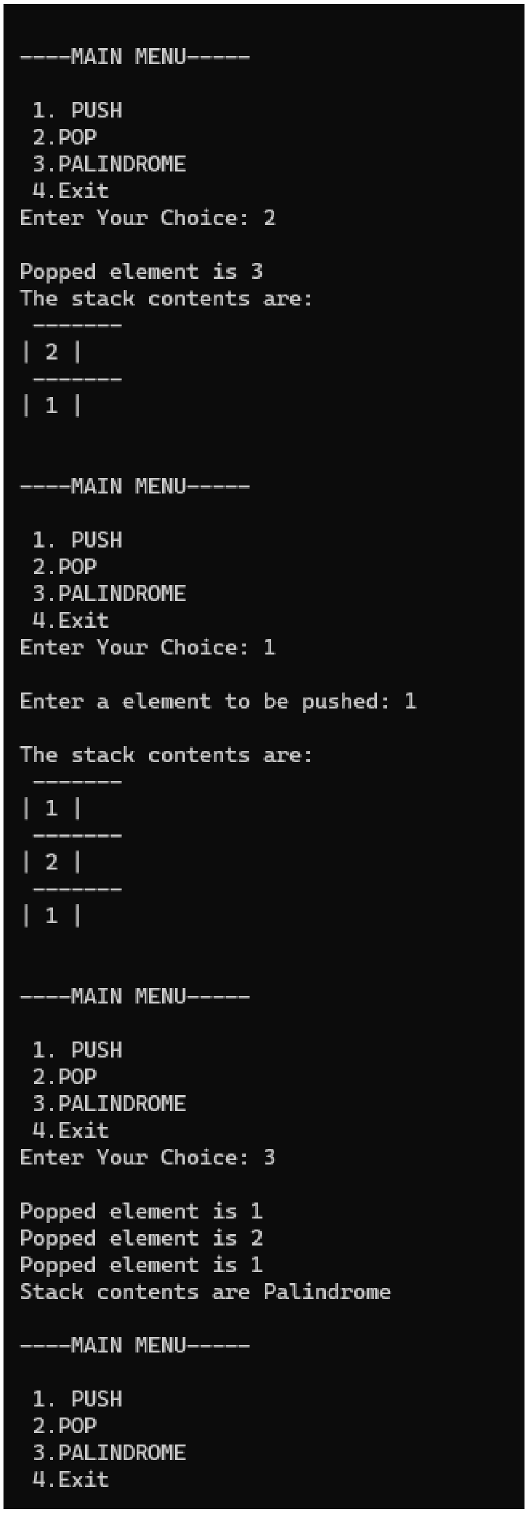
Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<conio.h>
#define MAX 4
int stack[MAX], item;
int ch, top = -1, count = 0, status = 0;
/*PUSH FUNCTION*/
void push(int stack[], int item){
if (top == (MAX-1))
printf("\n\nStack is Overflow");
else{
printf("\nEnter a element to be pushed: ");
scanf("%d", &item);
stack[++top] = item;
status++;
}
}
/*POP FUNCTION*/
int pop(int stack[]){
int ret;
if(top == -1)
printf("\n\nStack is Underflow");
else
{
ret = stack[top--];
status--;
}
printf("\nPopped element is %d", ret);
return ret;
}
void palindrome(int stack[]){
int i, temp;
temp = status;
for(i=0; i<temp; i++)
{
if(stack[i] == pop(stack))
count++;
}
if(temp==count)
printf("\nStack contents are Palindrome");
else
printf("\nStack contents are not palindrome");
}
/*FUNCTION TO DISPLAY STACK*/
void display(int stack[]){
int i;
printf("\nThe stack contents are:");
if(top == -1)
printf("\nStack is Empty");
else{
for(i=top; i>=0; i--)
printf("\n -------\n| %d |", stack[i]);
}printf("\n");
}
void main(){
do{
printf("\n\n----MAIN MENU-----\n");
printf("\n 1. PUSH\n 2.POP\n 3.PALINDROME\n 4.Exit ");
printf("\nEnter Your Choice: ");
scanf("%d", &ch);
switch(ch){
case 1:
push(stack, item);
display(stack);
break;
case 2:
item=pop(stack);
display(stack);
break;
case 3:
palindrome(stack);
break;
case 4:
exit(0);
break;
default:
printf("\nEND OF EXECUTION");
}
}while (ch != 4);
getch();
}
```

Output:



```
----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 1

Enter a element to be pushed: 1

The stack contents are:
 -------
| 1 |


----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 1

Enter a element to be pushed: 2

The stack contents are:
 -------
| 2 |
 -------
| 1 |


----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 1

Enter a element to be pushed: 3

The stack contents are:
 -------
| 3 |
 -------
| 2 |
 -------
| 1 |
```

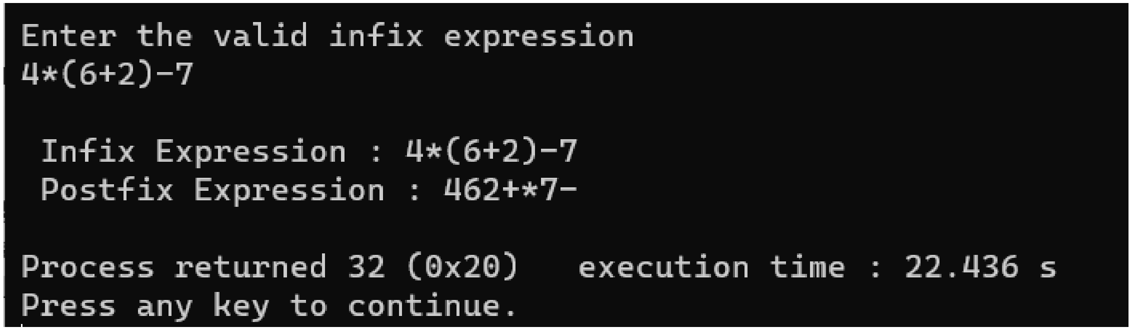


```
----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 2

Popped element is 3
The stack contents are:
 -------
| 2 |
 -------
| 1 |


----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 1

Enter a element to be pushed: 1

The stack contents are:
 -------
| 1 |
 -------
| 2 |
 -------
| 1 |


----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
Enter Your Choice: 3

Popped element is 1
Popped element is 2
Popped element is 1
Stack contents are Palindrome

----MAIN MENU-----

 1. PUSH
 2.POP
 3.PALINDROME
 4.Exit
```

4. Design, develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

```
#include<stdio.h>
void infix_to_postfix();
void push(char);
char pop();
int priority(char);
char infix[30],postfix[30],stack[30];
int top=-1;
void push(char item){
stack[++top]=item;
}
char pop(){
return stack[top--];
}
int priority(char symb){
int p;
switch(symb){
case '+':
case '-':p=1;break;
case '*':
case '/':
case '%': p=2;break;
case '^':p=3;break;
case '(':
case ')':p=0;break;
case '#':p=-1; // stack contain nothing
break;
}
return p;
}
void infix_to_postfix(){
int i=0,j=0;
char symb,temp;
push('#');
for(i=0;infix[i]!='\0';i++){
symb=infix[i];
switch(symb){
case '(' :
push(symb);
break;
case ')' :
temp=pop();
while(temp!='('){
postfix[j++] = temp;
temp=pop();
}
break;
case '+':
case '-':
case '*':
case '/':
case '%':
case '^':

while(priority (stack[top])>=priority(symb)){
temp = pop();
postfix[j++]=temp;
}
push(symb);
break;
default : postfix[j++] = symb;
}
}
while(top>0){
temp =pop();
postfix[j++] = temp;
} postfix[j] = '\0';
}
void main(){
printf("Enter the valid infix expression \n");
scanf("%s",infix);
infix_to_postfix();
printf("\n Infix Expression : %s",infix);
printf("\n Postfix Expression : %s \n",postfix);
}
```

Output:



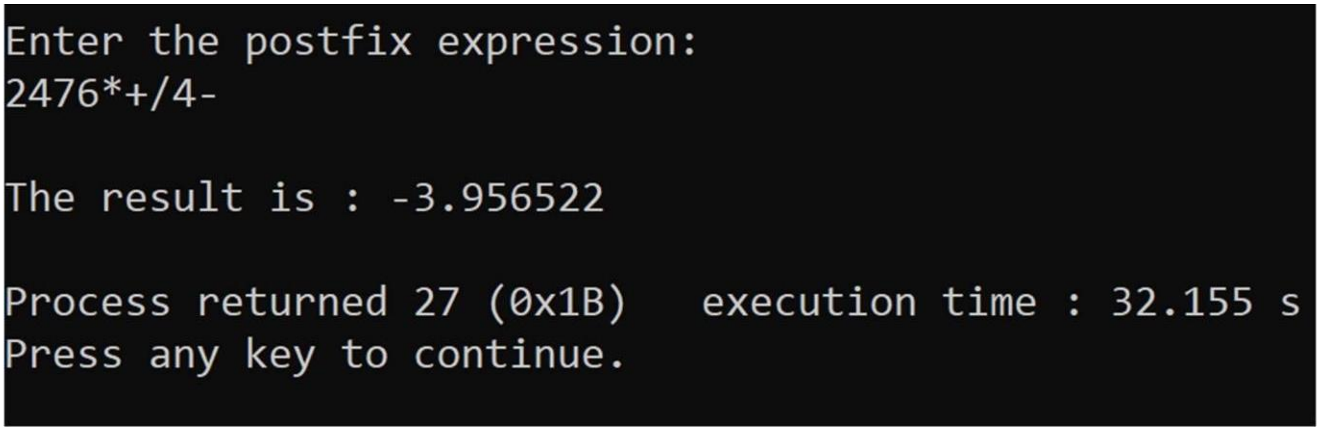
```
Enter the valid infix expression
4*(6+2)-7

 Infix Expression : 4*(6+2)-7
 Postfix Expression : 462+*7-

Process returned 32 (0x20)   execution time : 22.436 s
Press any key to continue.
```

5. Design, Develop and Implement a Program in C for the following Stack Applications. Evaluation of suffix expression with single digit operand and operators: +, -, *, /, %,^

```
#include<stdio.h>
#include<math.h>
#include<string.h>
float compute(char symbol, float op1, float op2)
{
   switch (symbol)
   {
   case '+':
      return op1 + op2;
   case '-':
      return op1 - op2;
   case '*':
      return op1 * op2;
   case '/':
      return op1 / op2;
   case '$':
   case '^':
      return pow(op1,op2);
   default :
      return 0;
   }
}
void main()
{
   float s[20], res, op1, op2;
   int top, i;
   char postfix[20], symbol;
   printf("\nEnter the postfix expression:\n");
   scanf ("%s", postfix);
   top=-1;
   for (i=0; i<strlen(postfix) ; i++)
   {
      symbol = postfix[i];
      if(isdigit(symbol))
         s[++top]=symbol - '0';
      else
      {
         op2 = s[top--];
         op1 = s[top--];
         res = compute(symbol,op1, op2);
         s[++top] = res;
      }
   }
   printf("\nThe result is : %f\n", res);
```

}

Output:

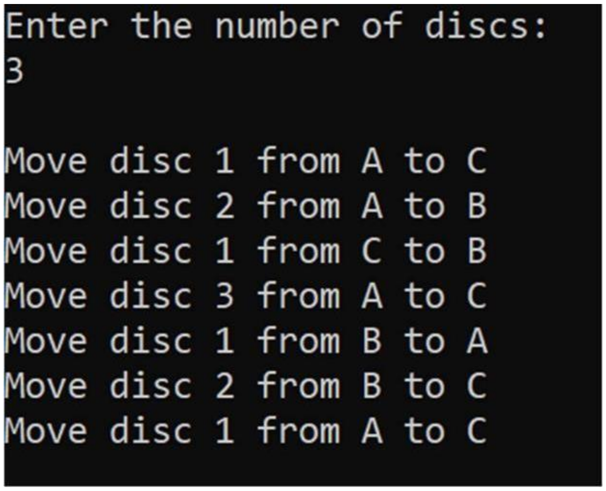


```
Enter the postfix expression:
2476*+/4-

The result is : -3.956522

Process returned 27 (0x1B)   execution time : 32.155 s
Press any key to continue.
```

6. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
#include<conio.h>
void tower(int n, int source, int temp,int destination)
{
   if(n == 0)
      return;
   tower(n-1, source, destination, temp);
   printf("\nMove disc %d from %c to %c", n, source, destination); // c= source tower to c=
destination
   tower(n-1, temp, source, destination); //n-1 means last disk
}
void main()
{
   int n; // number of disks
   printf("\nEnter the number of discs: \n");
   scanf("%d", &n);
   tower(n, 'A', 'B', 'C'); //it contains the n= number of disk, A,B,C : name of tower
   getch();
}
```

Output:

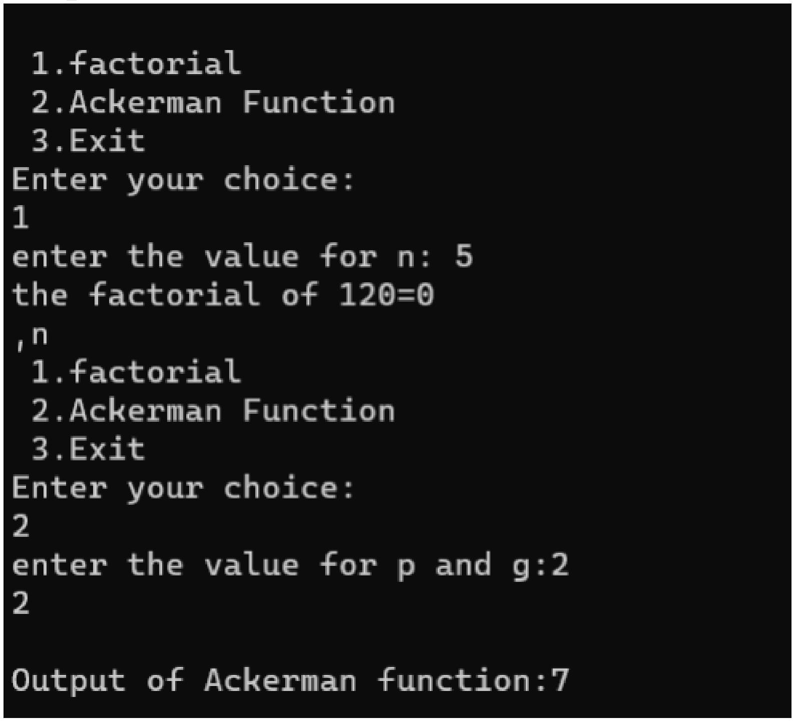


```
Enter the number of discs:
3

Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C
```

7. Program to implement factorial of a number and to generate the Ackerman function using recursive

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int fact(int n){
 if(n==0)
 return 1;
 return n*fact(n-1);
}
int A(int p,int q){
if(p==0)
 return q+1;
else if(q==0)
 return A(p-1,1);
else
 return A(p-1,A(p,q-1));
}
void main(){
int n,p,q,ch;
while(1){
 printf("\n 1.factorial\n 2.Ackerman Function\n 3.Exit\n");
 printf("Enter your choice:\n");
 scanf("%d",&ch);
 switch(ch) {
 case 1:printf("enter the value for n: ");
 scanf("%d",&n);
 printf("the factorial of %d=%d\n,n",fact(n));
 break;
 case 2:printf("enter the value for p and g:");
scanf("%d%d",&p,&q);
printf("\nOutput of Ackerman function:%d\n",A(p,q));
break;
case 3:exit(0);
default:printf("Invalid choice");
return;
}
}
}
```

Output:



```
 1.factorial
 2.Ackerman Function
 3.Exit
Enter your choice:
1
enter the value for n: 5
the factorial of 120=0
,n
 1.factorial
 2.Ackerman Function
 3.Exit
Enter your choice:
2
enter the value for p and g:2
2

Output of Ackerman function:7
```

8. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a) Insert an Element on to Circular QUEUE

b) Delete an Element from Circular QUEUE

c) Demonstrate Overflow and Underflow situations on Circular QUEUE

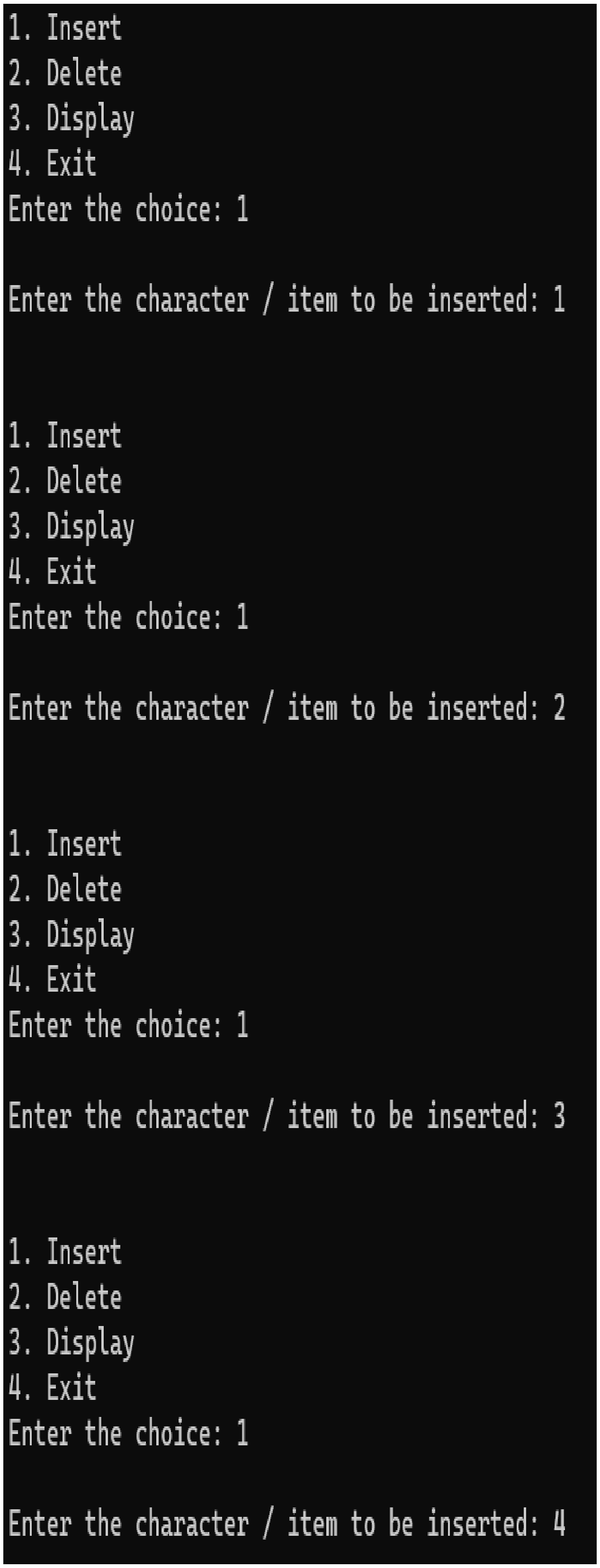
d) Display the status of Circular QUEUE

e) Exit

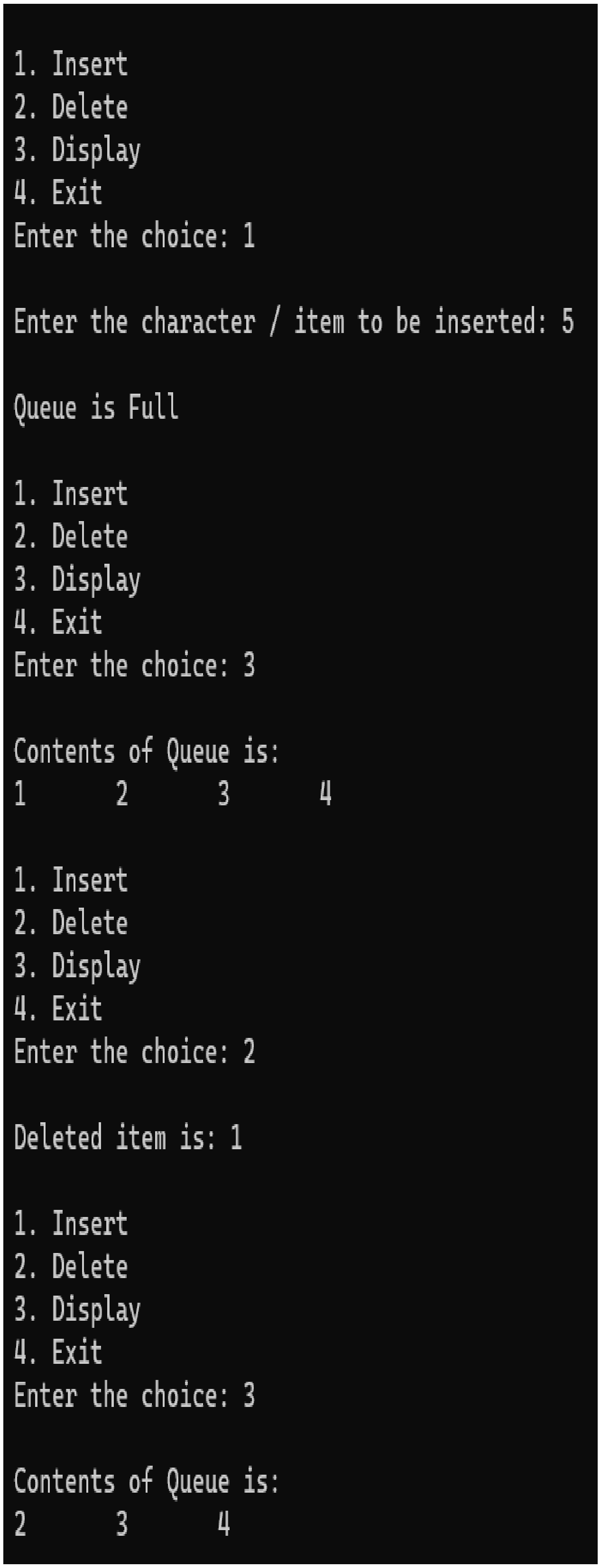
Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
int ch, front = 0, rear = -1, count=0;
char q[MAX], item;
void insert(){
if(count == MAX){
printf("\nQueue is Full");
} else {
rear = (rear + 1)% MAX;
q[rear]=item;
count++;
}
}
void del(){
if(count == 0)
printf("\nQueue is Empty");
else {
if(front > rear && rear==MAX-1)
{
front=0; rear=-1; count=0;
} else{
item=q[front];
printf("\nDeleted item is: %c",item);
front = (front + 1)% MAX;
count--;
}
}
}
void display(){
int i, f=front, r=rear;
if(count == 0)
printf("\nQueue is Empty");
else {
printf("\nContents of Queue is:\n");
for(i=f; i!=r; i=(i+1)% MAX) {
printf("%c\t", q[i]);
}
printf("%c\t", q[i]);
}
}
void main(){
do {
printf("\n\n1. Insert\n2. Delete\n3. Display\n4. Exit");
printf("\nEnter the choice: ");
scanf("%d", &ch);
switch(ch) {
case 1: printf("\nEnter the character / item to be inserted: ");
scanf("%s",&item);
insert();
break;
case 2: del();
break;
case 3: display();
break;
case 4: exit(0);
break;
}
}while(ch!=4);
getch();
}
```

Output:



```
1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 1

Enter the character / item to be inserted: 1


1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 1

Enter the character / item to be inserted: 2


1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 1

Enter the character / item to be inserted: 3


1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 1

Enter the character / item to be inserted: 4
```



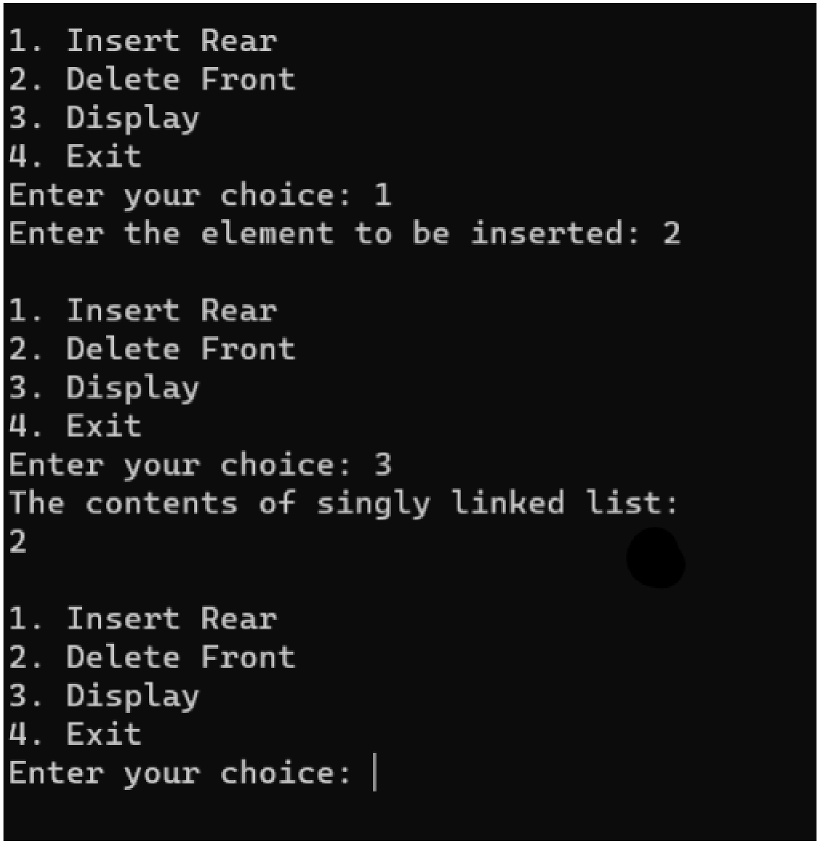
```

1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 1

Enter the character / item to be inserted: 5

Queue is Full

1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 3

Contents of Queue is:
1       2       3       4

1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 2

Deleted item is: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter the choice: 3

Contents of Queue is:
2       3       4
```

9. Program to implement singly Linked list using Queue

```
#include<stdio.h>
#include<stdlib.h>
// Node structure
struct node {
int info;
struct node *link;
};
// Function to create a new node
struct node* getnode() {
struct node* x = (struct node*)malloc(sizeof(struct node));
if(x == NULL) {
printf("Out of memory\n");
exit(0);
}
return x;
}
// Function to delete a node
void freenode(struct node *x) {
free(x);
}
// Function to insert a node at the rear end
struct node* insert_rear(int item, struct node *first) {
struct node *temp, *cur;
temp = getnode();
temp->info = item;
temp->link = NULL;
if(first == NULL) {
return temp;
}
cur = first;
while(cur->link != NULL) {
cur = cur->link;
}
cur->link = temp;
return first;
}
// Function to delete a node from the front end
struct node* delete_front(struct node *first) {
struct node *temp;
if(first == NULL) {
printf("List is empty, cannot delete\n");
return first;
}
temp = first;
first = first->link;
printf("Item deleted = %d\n", temp->info);
freenode(temp);
return first;
}
// Function to display the contents of the linked list
void display(struct node *first) {
struct node *temp;
if(first == NULL) {
printf("List is empty\n");
return;
}
printf("The contents of singly linked list:\n");
temp = first;
```

```
while(temp != NULL) {
printf("%d ", temp->info);
temp = temp->link;
}
printf("\n");
}
// Main function
int main() {
struct node *first = NULL;
int ch, item;
while(1) {
printf("\n1. Insert Rear\n2. Delete Front\n3. Display\n4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &ch);
switch(ch) {
case 1:
printf("Enter the element to be inserted: ");
scanf("%d", &item);
first = insert_rear(item, first);
break;
case 2:
first = delete_front(first);
break;
case 3:
display(first);
break;
case 4:
exit(0);
default:
printf("Invalid choice, please try again.\n");
}
}
return 0;
}
```

Output:

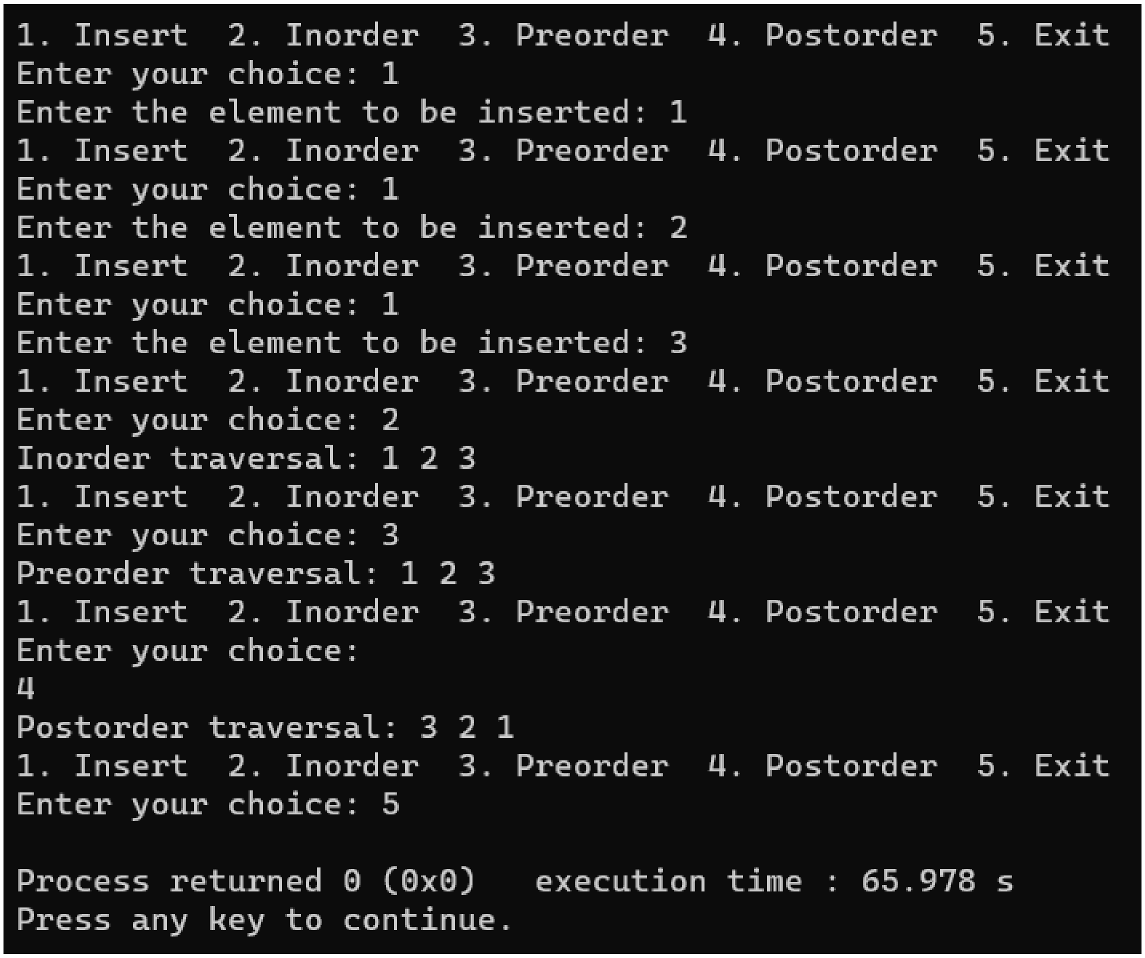


```
1. Insert Rear
2. Delete Front
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 2

1. Insert Rear
2. Delete Front
3. Display
4. Exit
Enter your choice: 3
The contents of singly linked list:
2

1. Insert Rear
2. Delete Front
3. Display
4. Exit
Enter your choice:
```

10 . Program to implement Binary tree traversal

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* createNode(int val) {
    struct Node* newNode = malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->left = newNode->right = NULL;
    return newNode;
}
void inorder(struct Node* root) {
    if (root) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
void preorder(struct Node* root) {
    if (root) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void postorder(struct Node* root) {
    if (root) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}
struct Node* insertNode(struct Node* node, int val) {
    if (!node)
        return createNode(val);
    if (val < node->data)
        node->left = insertNode(node->left, val);
    else if (val > node->data)
        node->right = insertNode(node->right, val);
    return node;
}
int main() {
    struct Node* root = NULL;
    int choice, item;
    while (1) {
        printf("1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the element to be inserted: ");
                scanf("%d", &item);
                root = insertNode(root, item);
                break;
            case 2:
                printf("Inorder traversal: ");
                inorder(root);
                printf("\n");
                break;
            case 3:
                printf("Preorder traversal: ");
                preorder(root);
                printf("\n");
                break;
            case 4:
                printf("Postorder traversal: ");
                postorder(root);
                printf("\n");
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Output:

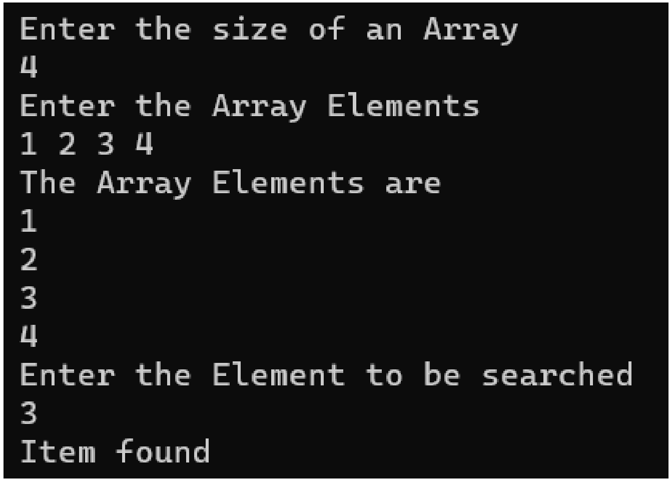


```
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 1
Enter the element to be inserted: 1
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 1
Enter the element to be inserted: 2
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 1
Enter the element to be inserted: 3
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 2
Inorder traversal: 1 2 3
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 3
Preorder traversal: 1 2 3
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice:
4
Postorder traversal: 3 2 1
1. Insert  2. Inorder  3. Preorder  4. Postorder  5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 65.978 s
Press any key to continue.
```

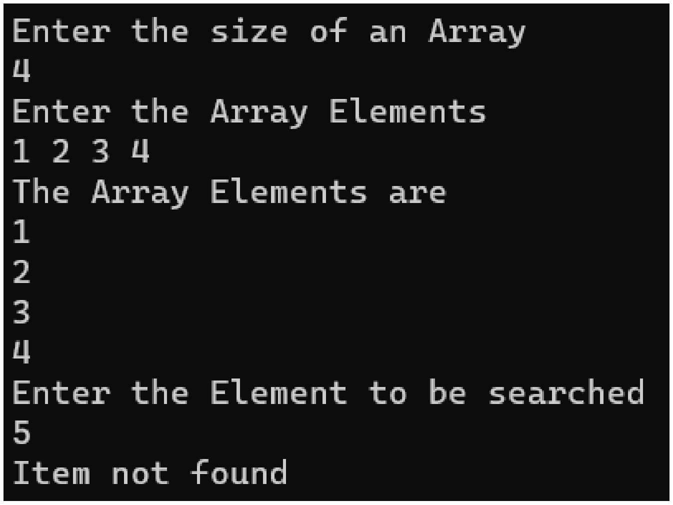
11. Program to implement Binary Search

```
#include<stdio.h>
#include<conio.h>
int search( int item, int a[ ], int n)
{
int low, high,key,mid;
low = 0; //Initialization
high = n-1; // Initialization
key=item;
while( low <= high )
{
mid = ( low + high ) / 2; // Find the mid-point
if ( key == a[mid] )
{
 // If item not found, return position
return mid;
}
 if( key < a[mid] )
 high = mid - 1; // Search left side
else
low = mid + 1; // Search right side
}
return -1; // Item not found
}
void main( )
{
 int i,item,a[10],n,pos;
 printf("Enter the size of an Array\n");
 scanf("%d",&n);
 printf("Enter the Array Elements\n");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
 printf("The Array Elements are\n");
 for(i=0;i<n;i++)
 printf("%d\n",a[i]);
 printf("Enter the Element to be searched\n");
 scanf("%d",&item);
 pos=search(item,a,n);
 if(pos==-1)
 printf("Item not found\n");
 else
 printf("Item found\n");
 getch( );
}
```

Output:



```
Enter the size of an Array
4
Enter the Array Elements
1 2 3 4
The Array Elements are
1
2
3
4
Enter the Element to be searched
3
Item found
```

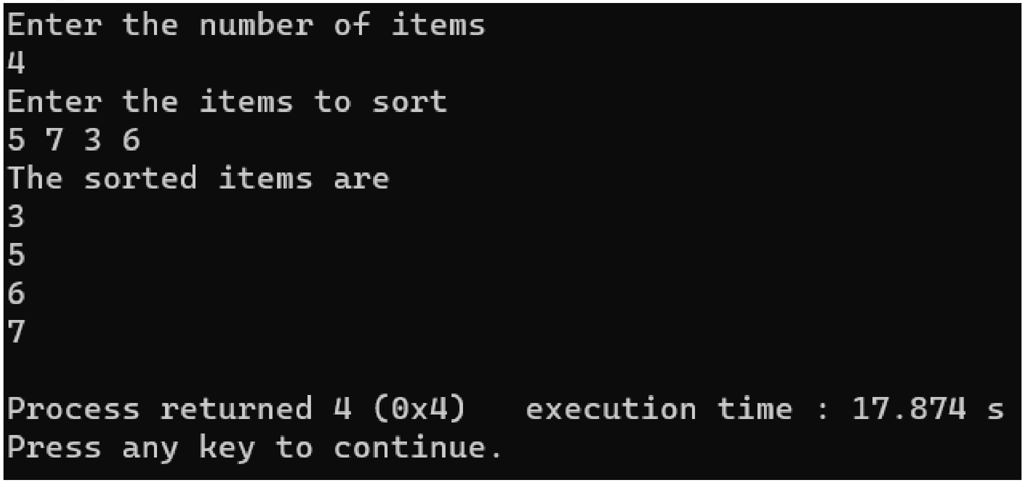


```
Enter the size of an Array
4
Enter the Array Elements
1 2 3 4
The Array Elements are
1
2
3
4
Enter the Element to be searched
5
Item not found
```

12. Implement bubble sort to sort a given array in c programming

```
#include<stdio.h>
void main( ){
int n,i,j,temp,a[20],pos;
printf(“Enter the number of items\n”);
scanf(“%d”,&n);
printf(“Enter the items to sort\n”);
for(i=0;i<n;i++)
sacnf(“%d”,&a[i]);
for(i=0;i<n-1;i++){
pos=i;
for(j=i+1;j<n;j++){
if(a[j]<a[pos])
pos=j;
}
temp=a[pos];
a[pos]=a[i];
a[i]=temp;
}
printf(“The sorted items are\n”);
for(i=0;i<n;i++)
printf(“%d\n”,a[i]);
}
```

Output:



```
Enter the number of items
4
Enter the items to sort
5 7 3 6
The sorted items are
3
5
6
7

Process returned 4 (0x4)   execution time : 17.874 s
Press any key to continue.
```