JADAVPUR UNIVERSITY

DATA STRUCTURE AND ALGORITHM

LAB ASSIGNMENTS

Name: SHRUTI PATHAK

Roll no: 002210503021

Dept: Computer Science and Engineering

Class: MCA 1st year 2nd sem

Session: 2022-2024

DSA ASSIGNMENT 1:

1. **Write a menu-driven program in C to perform the following operations in an integer array allocated memory dynamically. The list may grow or shrink compared to the initial allocation as and when required. Multiple operations may be performed on the existing list without recompiling/re‐executing the program. The individual operation should be implemented using a single function.**

List of operations:

a) Create
b) Display the entire list
c) Count (total number of elements in the list)
d) Reverse the list (reversed list should be stored in the same array)
e) Index of a given element (all indices if there are duplicate entries)
f) Indexed element
g) Insert
h) Delete
i) Merge
j) Split
k) Sort
l) Search

SOURCE CODE:

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

int* create(int *ptr,int n)
{
    int i;
    ptr=(int*)malloc(n*sizeof(int));
    printf("Enter %d values for the array: ",n);
    for(i=0;i<n;i++)
    scanf("%d",&ptr[i]);
    printf("Elements entered successfully\n");
    return ptr;
}

void display(int *ptr,int n)
{
    int i;
    if(n==0)
    printf("The array is empty!");
    else{
        printf("The values of the array: \n");
        for(i=0;i<n;i++)
        printf("%d ",ptr[i]);
        printf("\n");
    }
}
```

```c
void reverse(int *ptr,int n)
{
    int i,temp;if(n==0)
    printf("The array is empty!");
    else{

        for(i=0;i<n/2;i++)
        {
            temp=ptr[i];
            ptr[i]=ptr[n-1-i];
            ptr[n-1-i]=temp;
        }
        printf("\nAfter Reverse,\n");
        display(ptr,n);
    }

}

void index_of(int *ptr,int n,int m)
{
    int i;
    if(n==0)
    printf("The array is empty!");
    else{
        printf("%d is present in the array at index/indices: ",m);
        for(i=0;i<n;i++)
        {
            if(ptr[i]==m)
            printf("%d ",i);
        }
        printf("\n");
    }
}

int insert(int *ptr,int n,int m,int k)
{
    int i;
    if(k>n)
        printf("Insertion not possible!");

    else{
        ptr=(int*)realloc(ptr,sizeof(int));
        n++;
        for(i=n-1;i>=k;i--)
        {
            ptr[i]=ptr[i-1];
        }
        ptr[k]=m;
        printf("Element inserted successfully\n");
        display(ptr,n);
    }
    return n;

}

int delete_at(int *ptr,int n,int k)
{
    int i;
    if(k>=n)
    printf("Deletion not possible!");
    else{

        for(i=k;i<n;i++)
        {
            ptr[i]=ptr[i+1];
        }
        n--;
        printf("Element deleted successfully\n");
        display(ptr,n);
    }
    return n;

}

void merge()
{
    int i,j=0,*first,*second,l1,l2,result[100];
    printf("Enter length of first array: ");
    scanf("%d",&l1);
    first=create(first,l1);
    printf("Enter length of second array: ");
    scanf("%d",&l2);
    second=create(second,l2);

    for(i=0;i<l1;i++){
    result[i]=first[i];
    }
    for(i=0,j=l1;i<l2 && j<l1+l2;i++,j++)
    result[j]=second[i];

    printf("\nAfter merge, ");
    display(result,l1+l2);
}

void split()
{
    int l,*a,first[10],second[10],pos,i,k1=0,k2=0;
    printf("Enter length of the array: ");
    scanf("%d",&l);
    a=create(a,l);
    printf("Enter the position to split the array in to two arrays: ");
    scanf("%d",&pos);
    if(pos>=l){
        printf("Split cannot be possible!!!");
        return;
    }
    for(i=0;i<l;i++)
    {
        if(i<pos)
        first[k1++]=a[i];
        else
        second[k2++]=a[i];
    }
    printf("\nFirst array: \n");
    display(first,k1);
    printf("Second array: \n");
    display(second,k2);

}

void sort(int *ptr,int n)
{
    int i,j,temp;
    if(n==0)
    printf("The array is empty!");
    else{
        for(i=0;i<n-1;i++)
        {
            for(j=0;j<n-i-1;j++)
            {
                if(ptr[j]>ptr[j+1])
                {
                    temp=ptr[j];
                    ptr[j]=ptr[j+1];
                    ptr[j+1]=temp;
                }
            }
        }
        printf("After sorting,\n");
        display(ptr,n);
    }

}

void search(int *ptr,int n,int m)
{
    int i;
    if(n==0)
    printf("The array is empty!");
    else{
        printf("%d is present at index: ",m);
        for(i=0;i<n;i++)
        {
            if(ptr[i]==m)
            {
                printf("%d ",i);
                break;
            }
        }
        printf("\n");
    }
}

int main()
{
    int c,m,i,j,k,n;
    int *array;
    do{
        printf("\n----------MENU-------------\n");
        printf("0.Exit\n");
        printf("1.Create\n");
        printf("2.Display the entire list\n");
        printf("3.Count\n");
        printf("4.Reverse\n");
        printf("5.Index of a given element\n");
        printf("6.Indexed element\n");
        printf("7.Insert\n");
        printf("8.Delete\n");
        printf("9.Merge\n");
        printf("10.Split\n");
        printf("11.Sort\n");
        printf("12.Search\n");

        printf("\nEnter any choice: ");
        scanf("%d",&c);
        switch(c)
        {
            case 0:
                exit(0);
            case 1:
                printf("Enter number of elements: ");
                scanf("%d",&n);
                array=create(array,n);
                break;
            case 2:
                display(array,n);
                break;
            case 3:
                printf("The no of elements in the array is %d\n",n);
                break;
            case 4:
                reverse(array,n);
                break;
            case 5:
                printf("Enter the element whose index/indices you want to know: ");
                scanf("%d",&m);
                index_of(array,n,m);
                break;
            case 6:
                printf("Enter the index: ");
                scanf("%d",&i);
                if(i>=n)
                printf("Enter valid index! ");
                else
                printf("Element present at index %d is %d\n",i,array[i]);
                break;
            case 7:
                printf("Enter a element you want to insert: ");
                scanf("%d",&m);
                printf("Enter the position where you want to insert the element: ");
                scanf("%d",&k);
                n=insert(array,n,m,k);
                break;
            case 8:
                printf("Enter the position which you want to delete: ");
                scanf("%d",&k);
                n=delete_at(array,n,k);
                break;
            case 9:
                merge();
                break;
```

```
            case 10:
                split();
                break;
            case 11:
                sort(array,n);
                break;
            case 12:
                printf("Enter a element you want to search: ");
                scanf("%d",&m);
                search(array,n,m);
                break;
            default:
                printf("You have entered a wrong choice!!!\n");
        }


    }while(1);
    return 0;
}
```

output:

```
-----------MENU-------------
0.Exit
1.Create
2.Display the entire list
3.Count
4.Reverse
5.Index of a given element
6.Indexed element
7.Insert
8.Delete
9.Merge
10.Split
11.Sort
12.Search

Enter any choice: 1
Enter number of elements: 6
Enter 6 values for the array: 1 2 3 4 2 5
Elements entered successfully
```

```
Enter any choice: 2
The values of the array:
1 2 3 4 2 5
```

```
Enter any choice: 3
The no of elements in the array is 6
```

```
Enter any choice: 4

After Reverse,
The values of the array:
5 2 4 3 2 1
```

```
Enter any choice: 5
Enter the element whose index/indices you want to know: 2
2 is present in the array at index/indices: 1 4
```

```
Enter any choice: 6
Enter the index: 4
Element present at index 4 is 2
```

```
Enter any choice: 7
Enter a element you want to insert: 8
Enter the position where you want to insert the element: 3
Element inserted successfully
The values of the array:
5 2 4 8 3 2 1
```

```
Enter any choice: 8
Enter the position which you want to delete: 1
Element deleted successfully
The values of the array:
5 4 8 3 2 1
```

```
Enter any choice: 11
After sorting,
The values of the array:
1 2 3 4 5 8
```

```
Enter any choice: 12
Enter a element you want to search: 8

8 is found at position 6
```

```
Enter any choice: 9
Enter length of first array: 3
Enter 3 values for the array: 1 2 3
Elements entered successfully
Enter length of second array: 2
Enter 2 values for the array: 4 5
Elements entered successfully

After merge, The values of the array:
1 2 3 4 5

Enter any choice: 10
Enter length of the array: 7
Enter 7 values for the array: 4 7 1 8 9 2 3
Elements entered successfully
Enter the position to split the array in to two arrays: 3

First array:
The values of the array:
4 7 1
Second array:
The values of the array:
8 9 2 3
```

DSA ASSIGNMENT 3:

1. Write a program to implement a stack data structure using an array.

SOURCE CODE:

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

#define SIZE 100
int tos = -1;
int stack[SIZE];

void push(int val){
    if (tos==SIZE-1){
        printf("\nStack overflow!\n");
        return;
    }
    stack[++tos] = val;
    printf("\n%d is pushed into stack.\n",val);
}
```

```c
int pop(){
    if (tos==-1){
        printf("\nStack underflow!\n");
        return -1;
    }
    return stack[tos--];
}

void display()
{
    int i;
    if (tos==-1){
        printf("\nStack is empty!\n");
        return;
    }
    else{
        for(i=tos;i>=0;i--)
            printf("\n\t|\t%d\t|",stack[i]);
        printf("\n\t-----------------\n");

    }
}

int main(){
    int ch,val;
    do{
        printf("\n------------MENU FOR STACK----------------\n");
        printf("0.Exit\n");
        printf("1.PUSH\n");
        printf("2.POP\n");
        printf("3.DISPLAY\n");
        printf("------------------------------------------\n");
        printf("\nEnter any choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\nEnter a value: ");
                scanf("%d",&val);
                push(val);
                break;
            case 2:
                val=pop();
                if(val!=-1)
                printf("\n%d is popped from stack.\n",val);
                break;
            case 3:
                display();
                break;
            case 0:
                exit(0);
            default:
                printf("\nInvalid Choice!!!\n");
        }
    }while(1);

    return 0;
}
```

output:

```
------------MENU FOR STACK------------
0.Exit
1.PUSH
2.POP
3.DISPLAY
--------------------------------------

Enter any choice: 1

Enter a value: 5

5 is pushed into stack.
```

```
Enter any choice: 1

Enter a value: 3

3 is pushed into stack.
```

```
Enter any choice: 1

Enter a value: 6

6 is pushed into stack.
```

```
Enter any choice: 1

Enter a value: 2

2 is pushed into stack.
```

```
Enter any choice: 3

        |       2       |
        |       6       |
        |       3       |
        |       5       |
        -----------------
```

```
Enter any choice: 2

2 is popped from stack.
```

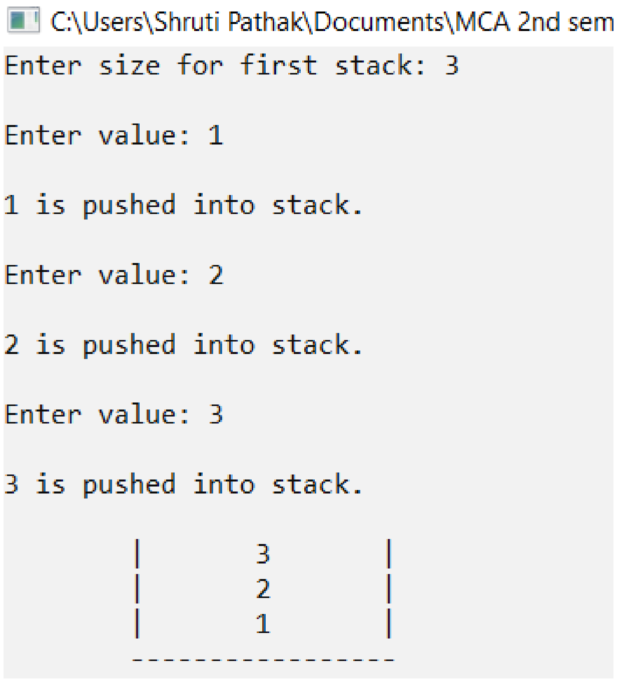
```
Enter any choice: 3

        |       6       |
        |       3       |
        |       5       |
        -----------------
```

4. Write a Boolean function to return true if two stacks are equal.

SOURCE CODE:

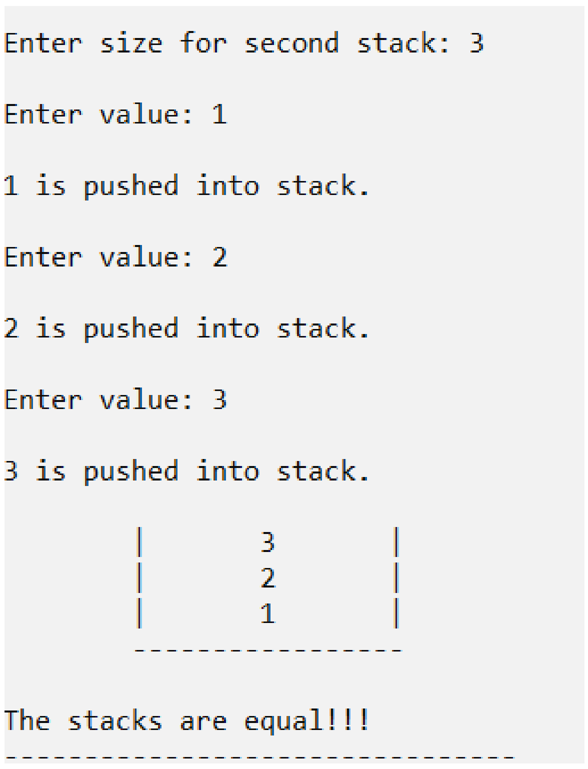
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int *data;
    int tos;
    int size;
}stack;

stack* create(int s){
    stack *st=(stack *)malloc(sizeof(stack));
    st->data=(int *)malloc(sizeof(int)*s);
    st->tos=-1;
    st->size=s;
    return st;
}

void push(stack *s,int val){
    if (s->tos==s->size-1){
        printf("\nStack overflow!\n");
        return;
    }
    s->data[++s->tos] = val;
    printf("\n%d is pushed into stack.\n",val);
}

void display(stack *s)
{
    int i;
    if (s->tos==-1){
        printf("\nStack is empty!\n");
        return;
    }
    else{
        for(i=s->tos;i>=0;i--)
            printf("\n\t|\t%d\t|",s->data[i]);
        printf("\n\t-----------------\n");

    }
}

bool is_equal(stack *a,stack *b)
{
    int i,j=0;
    if(a->size!=b->size)
        return false;
    for(i=0;i<=a->tos;i++)
    {
        if(a->data[i]!=b->data[j++]){
            return false;
        }
    }
    return true;
}

int main()
{
    stack *a,*b;
    int size1,size2,i,val;
    printf("Enter size for first stack: ");
    scanf("%d",&size1);
    a=create(size1);
    for(i=1;i<=size1;i++)
    {
        printf("\nEnter value: ");
        scanf("%d",&val);
        push(a,val);
    }
    display(a);

    printf("\nEnter size for second stack: ");
    scanf("%d",&size2);
    b=create(size2);
    for(i=1;i<=size2;i++)
    {
        printf("\nEnter value: ");
        scanf("%d",&val);
        push(b,val);
    }
    display(b);

    if(is_equal(a,b))
    printf("\nThe stacks are equal!!!");
    else
    printf("\nThe stacks are not equal!!!");
}
```

output:

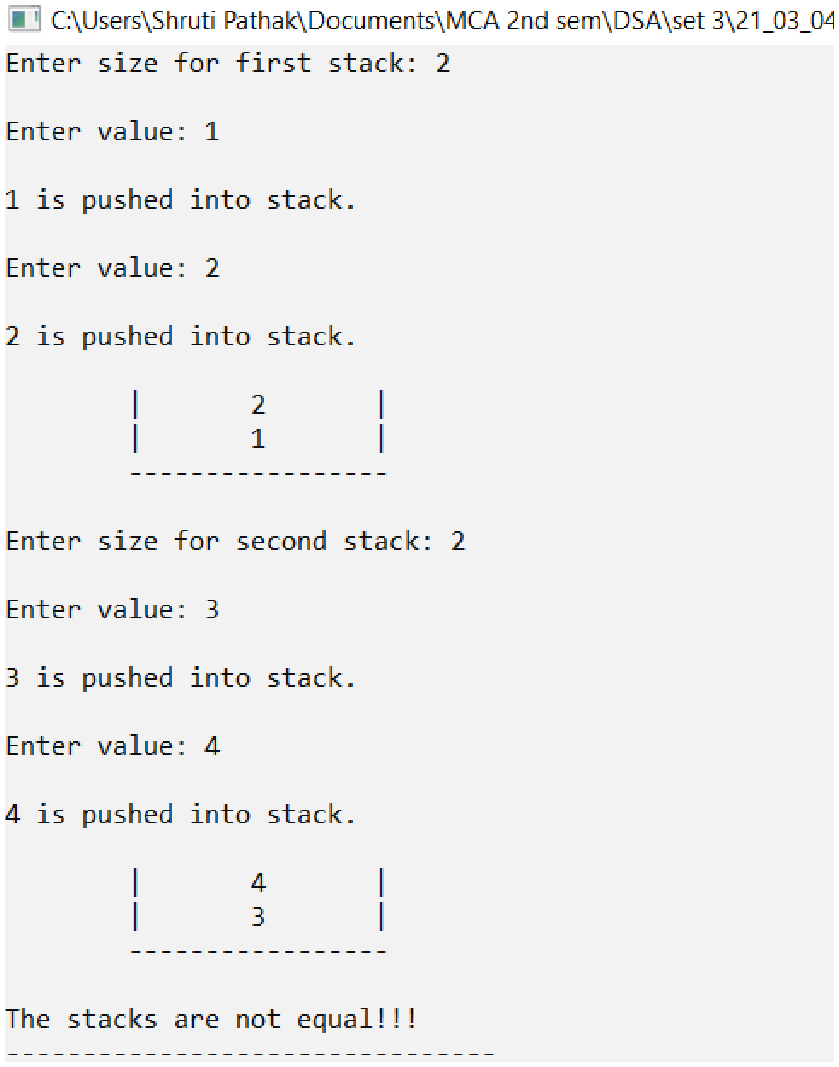


C:\Users\Shruti Pathak\Documents\MCA 2nd sem

```
Enter size for first stack: 3

Enter value: 1

1 is pushed into stack.

Enter value: 2

2 is pushed into stack.

Enter value: 3

3 is pushed into stack.

        |       3       |
        |       2       |
        |       1       |
        -----------------
```



```
Enter size for second stack: 3

Enter value: 1

1 is pushed into stack.

Enter value: 2

2 is pushed into stack.

Enter value: 3

3 is pushed into stack.

        |       3       |
        |       2       |
        |       1       |
        -----------------

The stacks are equal!!!
--------------------------------
```



C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 3\21_03_04

```
Enter size for first stack: 2

Enter value: 1

1 is pushed into stack.

Enter value: 2

2 is pushed into stack.

        |       2       |
        |       1       |
        -----------------

Enter size for second stack: 2

Enter value: 3

3 is pushed into stack.

Enter value: 4

4 is pushed into stack.

        |       4       |
        |       3       |
        -----------------

The stacks are not equal!!!
--------------------------------
```

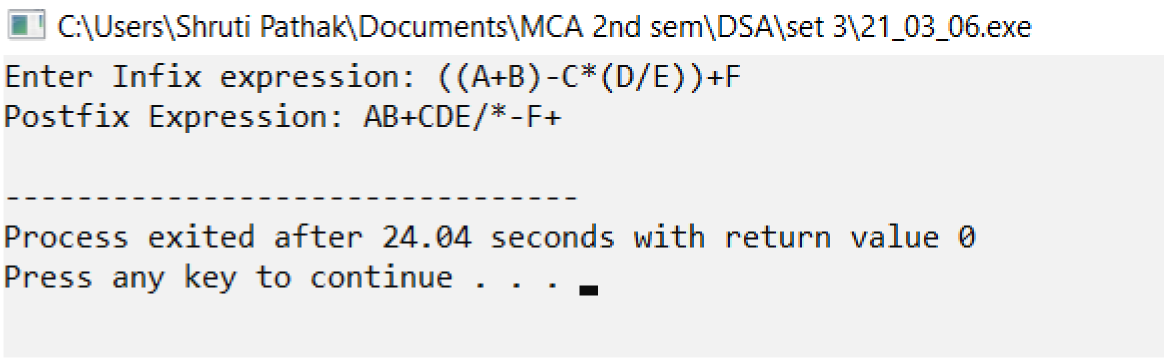
6. Using a stack, write a program to convert an infix expression into its equivalent postfix expression.

SOURCE CODE:

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

#define SIZE 100
int tos = -1;
char stack[SIZE];

void push(char val){
    if (tos==SIZE-1){
        printf("\nStack overflow!\n");
        return;
    }
    stack[++tos] = val;
}

char pop(){
    if (tos==-1){
        printf("\nStack underflow! Invalid infix expression\n");
        exit(1);
    }
    return stack[tos--];
}

int is_operator(char c)
{
    if(c=='^' || c=='*' || c=='/' || c=='+' || c=='-' )
    return 1;
    else
    return 0;
}

int precedence(char c)
{
    if(c=='^')
    return 3;
    else if(c=='*' || c=='/')
    return 2;
    else if(c=='+' || c=='-' )
    return 1;
    else
    return 0;
}

void InfixToPostfix(char infix[],char postfix[])
{
    int i=0,j=0;
    char item;
    char c;
    push('(');
```

```c
    strcat(infix,")");
    item=infix[i];
    while(item != '\0')
    {
        if(item=='(')
        push(item);
        else if(isdigit(item) || isalpha(item))
        postfix[j++]=item;
        else if(is_operator(item))
        {
            c=pop();
            while(is_operator(c)==1 && precedence(c) >= precedence(item)){
                postfix[j++]=c;
                c=pop();
            }
            push(c);
            push(item);
        }
        else if(item==')')
        {
            c=pop();
            while(c!='(')
            {
                postfix[j++]=c;
                c=pop();
            }
        }
        else{
            printf("Invalid infix expression!!!");
            exit(1);
        }
        i++;
        item=infix[i];
    }

    if(tos>0)
    {
        printf("Invalid infix expression!!!");
        exit(1);
    }
    postfix[j]='\0';
}

int main()
{
    char infix[SIZE],postfix[SIZE];
    printf("Enter Infix expression: ");
    gets(infix);
    InfixToPostfix(infix,postfix);
    printf("Postfix Expression: ");
    puts(postfix);
    return 0;
}
```

output:


C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 3\21_03_06.exe

```
Enter Infix expression: ((A+B)-C*(D/E))+F
Postfix Expression: AB+CDE/*-F+

--------------------------------
Process exited after 24.04 seconds with return value 0
Press any key to continue . . .
```

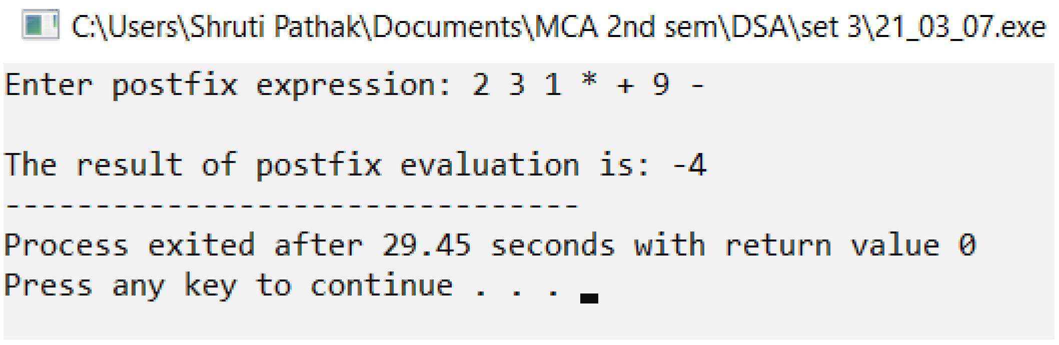
7. Write a program to evaluate postfix expression using a stack.

SOURCE CODE:

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

#define SIZE 100
int tos = -1;
char stack[SIZE];

void push(int val){
    if (tos==SIZE-1){
        printf("\nStack overflow!\n");
        return;
    }
    stack[++tos] = val;
}

int pop(){
    if (tos==-1){
        printf("\nStack underflow! Invalid postfix expression\n");
        exit(1);
    }
    return stack[tos--];
}

void PostfixEvaluation(char postfix[])
{
    int i,val,a,b;
    char item;
    for(i=0;postfix[i]!='\0';i++)
    {
        item=postfix[i];
        if(isdigit(item))
            push(item-'0');
        else if(item=='+' || item=='-' || item=='*' || item=='/')
        {
            a=pop();
            b=pop();
            switch(item)
```

```c
            {
                case '+':
                    val=b+a;
                    break;
                case '-':
                    val=b-a;
                    break;
                case '*':
                    val=b*a;
                    break;
                case '/':
                    val=b/a;
                    break;
            }
            push(val);
        }
    }
    if(tos>0)
    {
        printf("\nInvalid postfix expression!!!");
        exit(1);
    }
    printf("\nThe result of postfix evaluation is: %d",val);
}
int main()
{
    char postfix[SIZE];
    printf("Enter postfix expression: ");
    gets(postfix);
    PostfixEvaluation(postfix);
    return 0;
}
```

output:



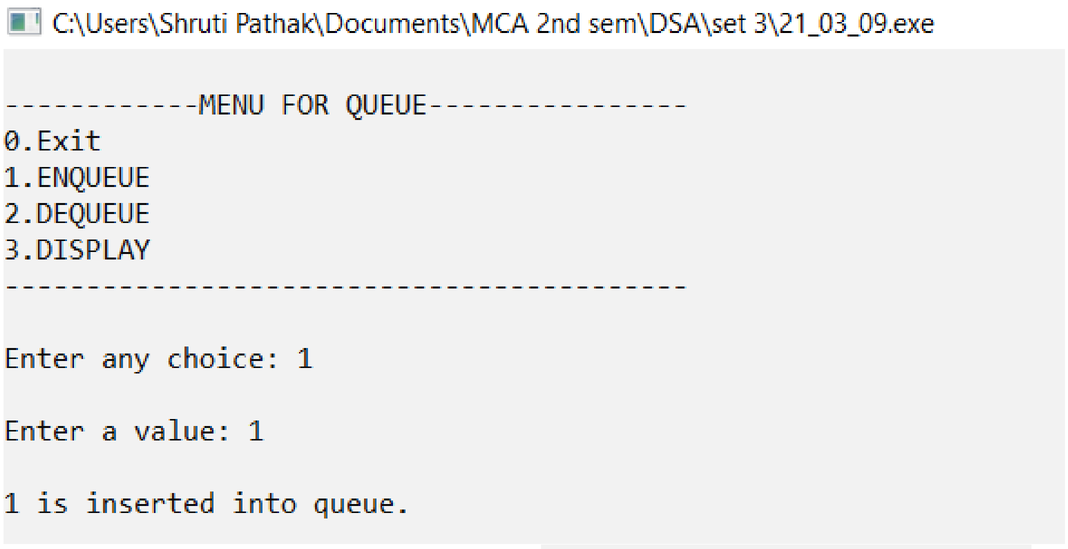
```
C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 3\21_03_07.exe
Enter postfix expression: 2 3 1 * + 9 -

The result of postfix evaluation is: -4
--------------------------------
Process exited after 29.45 seconds with return value 0
Press any key to continue . . .
```

9. Write a program to implement queue data structure using an array.

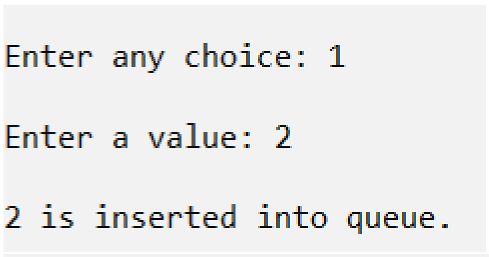
SOURCE CODE:

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

#define SIZE 100
int rear = -1;
int front = -1;

int queue[SIZE];

void enqueue(int val){
    if (rear==SIZE-1){
        printf("\nQueue overflow!\n");
        return;
    }
    if(front==-1)
    front=0;
    queue[++rear]=val;
    printf("\n%d is inserted into queue.\n",val);
}

int dequeue(){
    if (front==-1 || front>rear){
        printf("\nQueue underflow!\n");
        return -1;
    }
    return queue[front++];
}
void display()
{
    int i;
    if (front==-1 || front>rear){
        printf("\nQueue is empty!\n");
        return;
    }
    else{
        printf("\nQueue is: ");
        for(i=front;i<=rear;i++)
            printf("%d ",queue[i]);
        printf("\n");
    }
}

int main(){
    int ch,val;
    do{
        printf("\n------------MENU FOR QUEUE----------------\n");
        printf("0.Exit\n");
        printf("1.ENQUEUE\n");
        printf("2.DEQUEUE\n");
        printf("3.DISPLAY\n");
        printf("------------------------------------------\n");
        printf("\nEnter any choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\nEnter a value: ");
                scanf("%d",&val);
                enqueue(val);
                break;
            case 2:
                val=dequeue();
                if(val!=-1)
                printf("\n%d is deleted from queue.\n",val);
                break;
            case 3:
                display();
                break;
            case 0:
                exit(0);
            default:
                printf("\nInvalid Choice!!!\n");
        }
    }while(1);

    return 0;
}
```

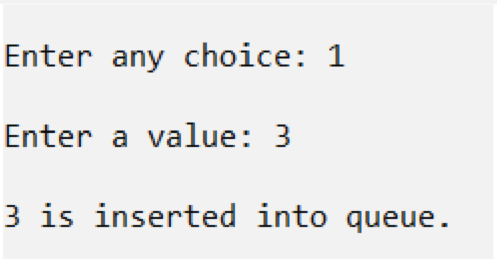
output:



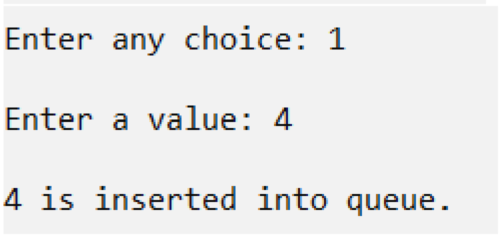
```
C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 3\21_03_09.exe

------------MENU FOR QUEUE----------------
0.Exit
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
------------------------------------------

Enter any choice: 1

Enter a value: 1

1 is inserted into queue.
```



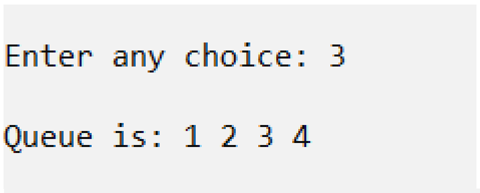
```
Enter any choice: 1

Enter a value: 2

2 is inserted into queue.
```



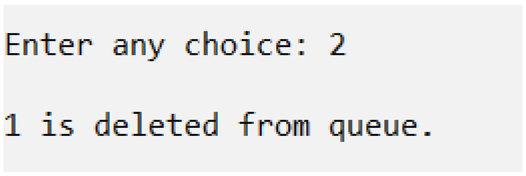
```
Enter any choice: 1

Enter a value: 3

3 is inserted into queue.
```



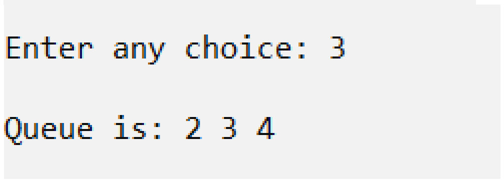
```
Enter any choice: 1

Enter a value: 4

4 is inserted into queue.
```



```
Enter any choice: 3

Queue is: 1 2 3 4
```



```
Enter any choice: 2

1 is deleted from queue.
```



```
Enter any choice: 3

Queue is: 2 3 4
```

DSA ASSIGNMENT 4:

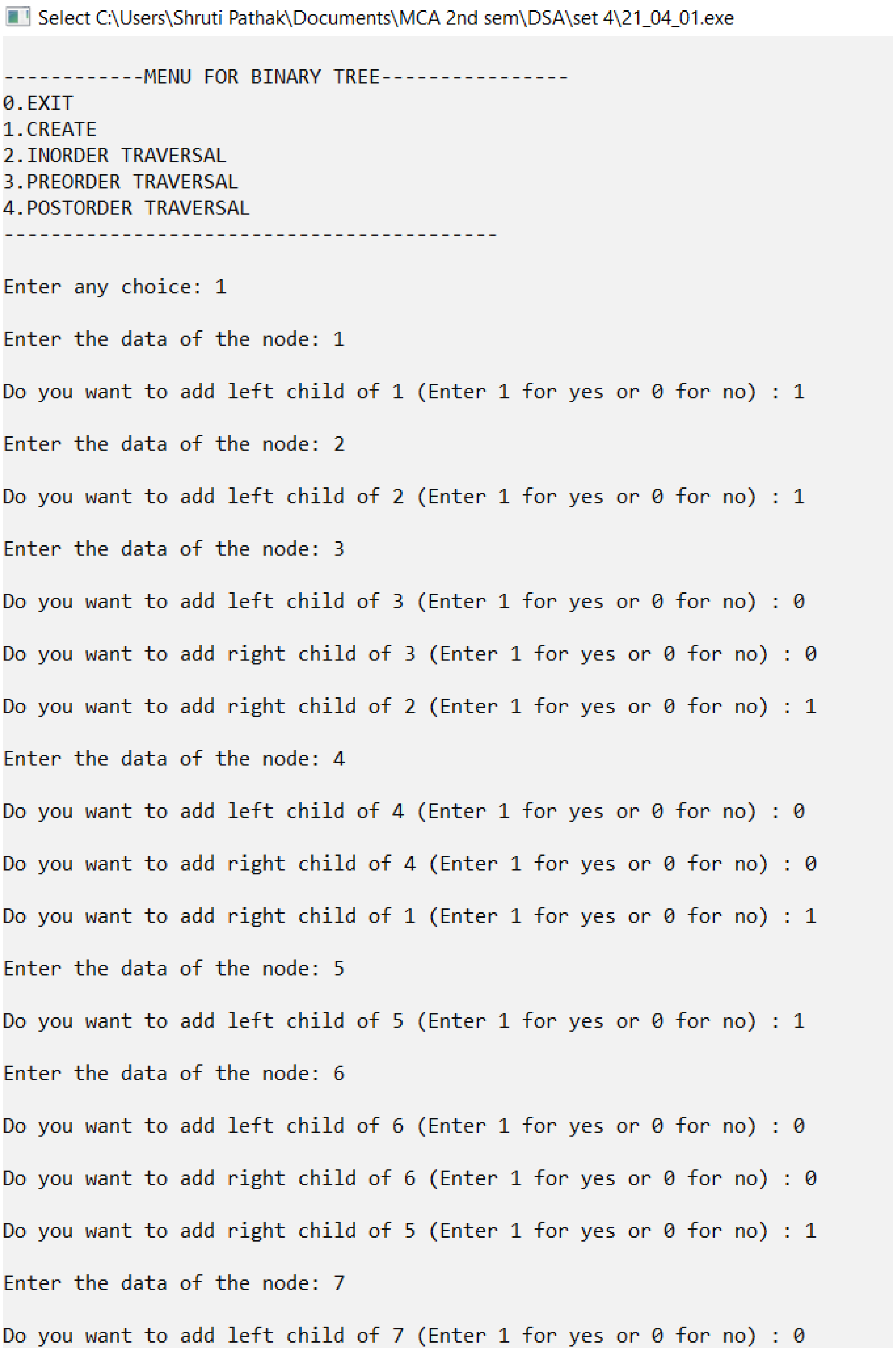
1. Write a menu-driven program for a binary tree using linked representation to (a)Create (b) Preorder traversal (c) Inorder traversal (d) Postorder traversal

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int data;
    struct node* left;
    struct node* right;
    }node;

void inorder(node *root)
{
    if(root==NULL)
    return;
    inorder(root->left);
    printf("%d ",root->data);
    inorder(root->right);
}

void preorder(node *root)
{
    if(root==NULL)
    return;
    printf("%d ",root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(node *root)
{
    if(root==NULL)
    return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ",root->data);
}
node* create()
{
    node *t;
    int ch;
    t=(node *)malloc(sizeof(node));
    printf("\nEnter the data of the node: ");
    scanf("%d",&t->data);
    printf("\nDo you want to add left child of %d (Enter 1 for yes or 0 for no) : " ,t-
>data);
    scanf("%d",&ch);
    if(ch)
        t->left=create();
    else
        t->left=NULL;
```

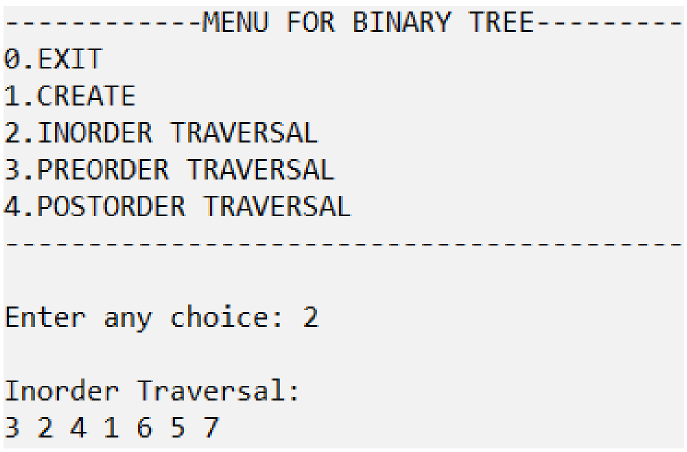
```c
    printf("\nDo you want to add right child of %d (Enter 1 for yes or 0 for no) : ",t-
>data);
    scanf("%d",&ch);
    if(ch)
        t->right=create();
    else
        t->right=NULL;
    return t;
}

int main(){
    int ch,val;
    node* root;
    do{
        printf("\n------------MENU FOR BINARY TREE----------------\n");
        printf("0.EXIT\n");
        printf("1.CREATE\n");
        printf("2.INORDER TRAVERSAL\n");
        printf("3.PREORDER TRAVERSAL\n");
        printf("4.POSTORDER TRAVERSAL\n");
        printf("------------------------------------------\n");
        printf("\nEnter any choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                root=create(root);
                break;
            case 2:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nInorder Traversal: \n");
                inorder(root);
                break;
            case 3:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nPreorder Traversal: \n");
                preorder(root);
                break;
            case 4:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nPostorder Traversal: \n");
                postorder(root);
                break;
            case 0:
                printf("Program terminates...\n");
                exit(0);
            default:
                printf("\nInvalid Choice!!!\n");
        }
    }while(1);
    return 0;
}
```

output:

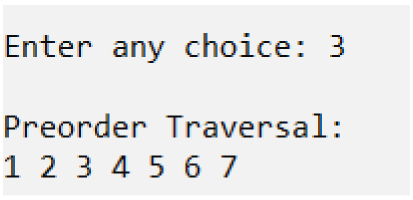


Select C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 4\21_04_01.exe

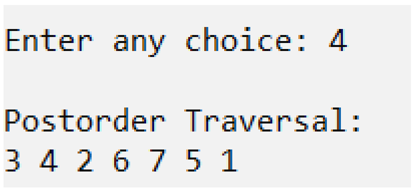
```
------------MENU FOR BINARY TREE----------------
0.EXIT
1.CREATE
2.INORDER TRAVERSAL
3.PREORDER TRAVERSAL
4.POSTORDER TRAVERSAL
------------------------------------------

Enter any choice: 1

Enter the data of the node: 1

Do you want to add left child of 1 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 2

Do you want to add left child of 2 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 3

Do you want to add left child of 3 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 3 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 2 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 4

Do you want to add left child of 4 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 4 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 1 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 5

Do you want to add left child of 5 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 6

Do you want to add left child of 6 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 6 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 5 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 7

Do you want to add left child of 7 (Enter 1 for yes or 0 for no) : 0
```



```
------------MENU FOR BINARY TREE---------
0.EXIT
1.CREATE
2.INORDER TRAVERSAL
3.PREORDER TRAVERSAL
4.POSTORDER TRAVERSAL
-----------------------------------------

Enter any choice: 2

Inorder Traversal:
3 2 4 1 6 5 7
```



```
Enter any choice: 3

Preorder Traversal:
1 2 3 4 5 6 7
```



```
Enter any choice: 4

Postorder Traversal:
3 4 2 6 7 5 1
```

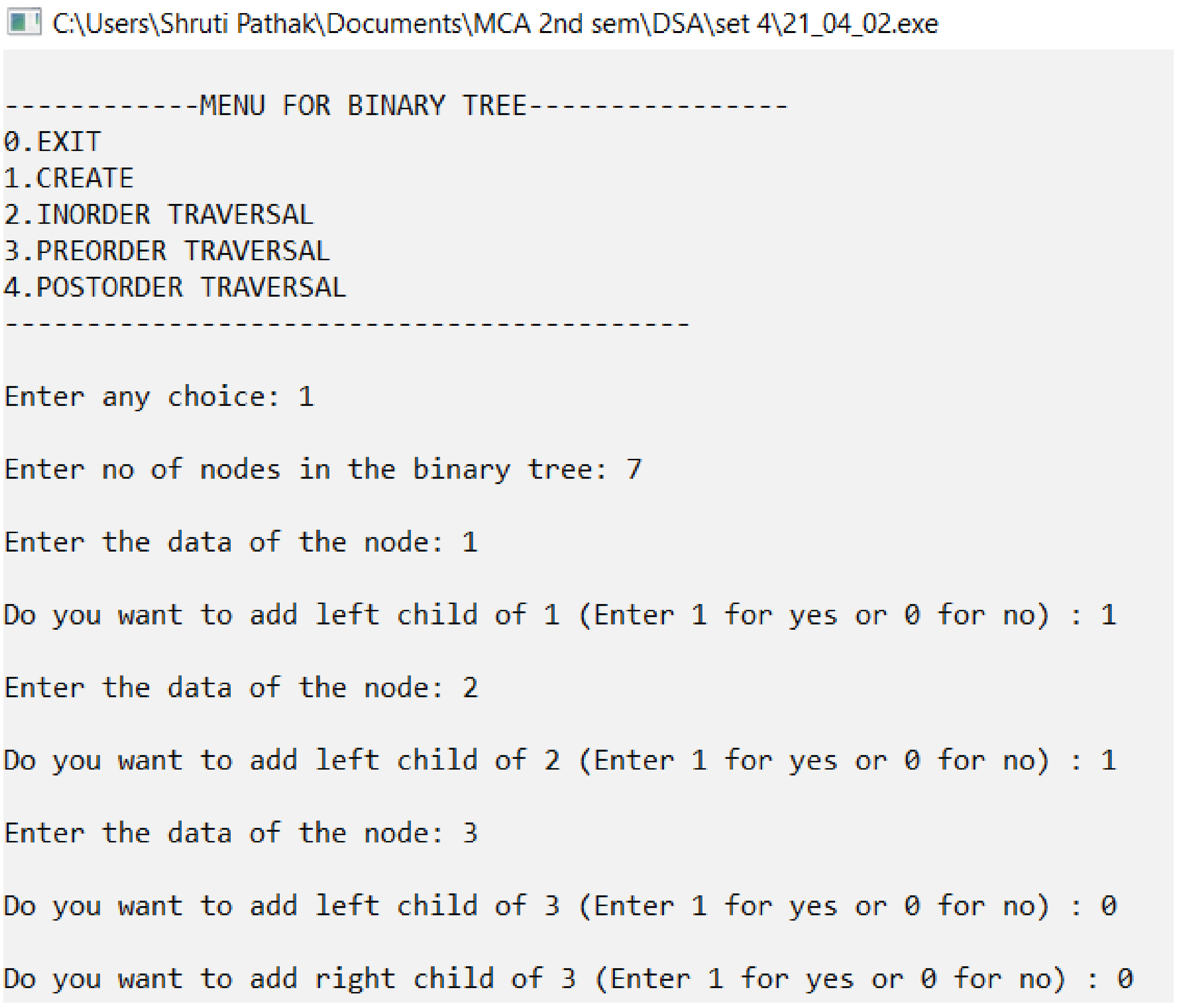
2. Write a menu-driven program for a binary tree using an array to (a)Create (b) Preorder traversal (c) Inorder traversal (d) Postorder traversal

SOURCE CODE:

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

void create(int *a,int n,int i)
{
    int d,ch;
    if(i<n)
    {
        printf("\nEnter the data of the node: ");
        scanf("%d",&d);
        a[i]=d;
        printf("\nDo you want to add left child of %d (Enter 1 for yes or 0 for no) :
",a[i]);
        scanf("%d",&ch);
        if(ch)
            create(a,n,2*i+1);
        printf("\nDo you want to add right child of %d (Enter 1 for yes or 0 for no) :
",a[i]);
        scanf("%d",&ch);
        if(ch)
            create(a,n,2*i+2);
    }
    else
        printf("Array is full\n");
    return;
}

void preorder(int *a,int n,int i)
{
    if(i>=n)
```

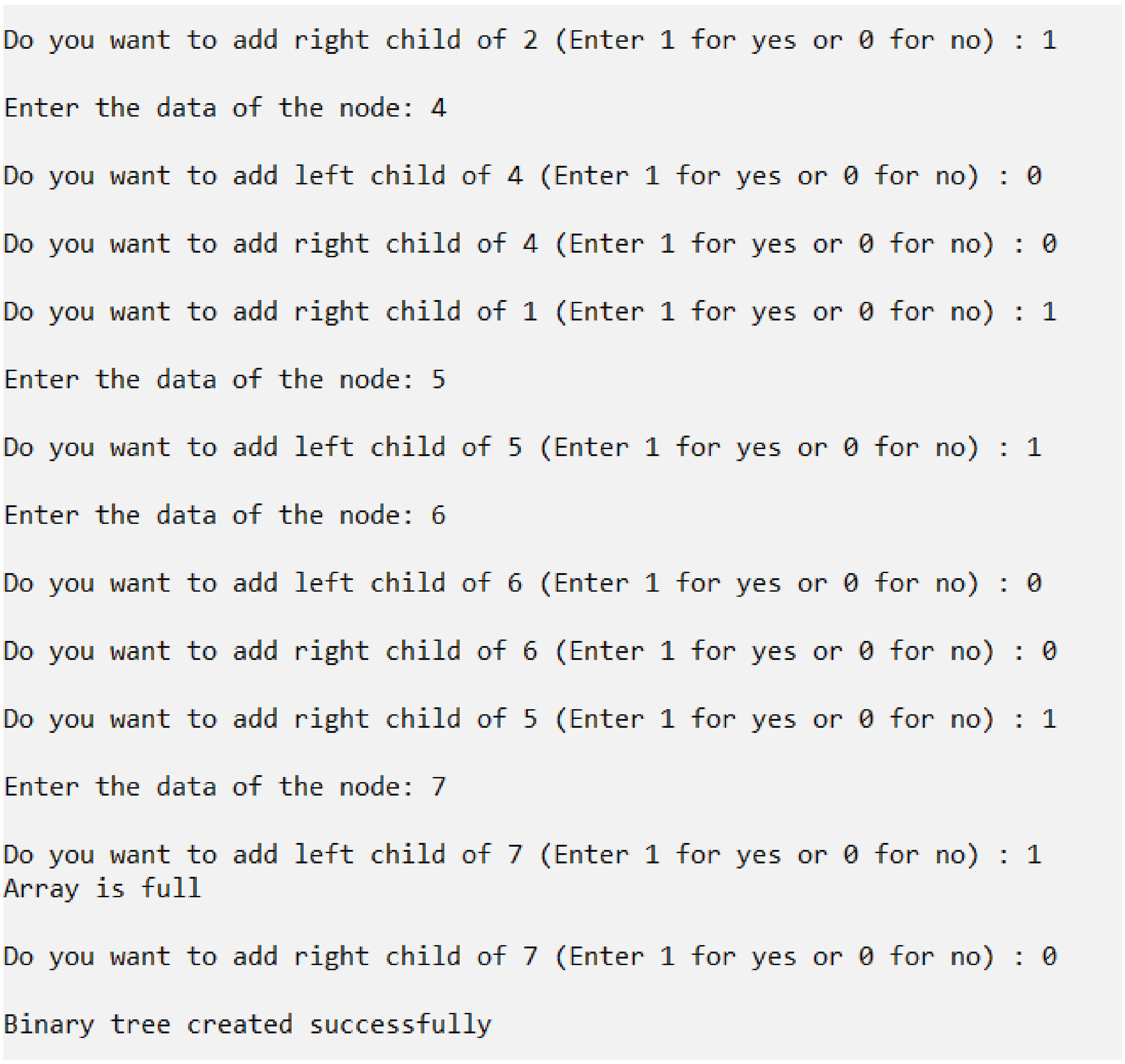
```
        return;
    printf("%d  ",a[i]);
    preorder(a,n,2*i+1);
    preorder(a,n,2*i+2);
    return;
}

void inorder(int *a,int n,int i)
{
    if(i>=n)
        return;
    inorder(a,n,2*i+1);
    printf("%d  ",a[i]);
    inorder(a,n,2*i+2);
    return;
}

void postorder(int *a,int n,int i)
{
    if(i>=n)
        return;
    postorder(a,n,2*i+1);
    postorder(a,n,2*i+2);
    printf("%d  ",a[i]);
    return;
}

int main(){
    int ch,val,n,i;
    int *root= NULL;
    do{
        printf("\n------------MENU FOR BINARY TREE----------------\n");
        printf("0.EXIT\n");
        printf("1.CREATE\n");
        printf("2.INORDER TRAVERSAL\n");
        printf("3.PREORDER TRAVERSAL\n");
        printf("4.POSTORDER TRAVERSAL\n");
        printf("------------------------------------------\n");
        printf("\nEnter any choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\nEnter no of nodes in the binary tree: ");
                scanf("%d",&n);
                root=(int *)malloc(sizeof(int)*n);
                create(root,n,0);
                printf("\nBinary tree created successfully\n");
                break;
            case 2:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nInorder Traversal: \n");
                inorder(root,n,0);
                break;
            case 3:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nPreorder Traversal: \n");
                preorder(root,n,0);
                break;
            case 4:
                if(root==NULL)
                printf("\nTree is empty!!!\n");
                else
                printf("\nPostorder Traversal: \n");
                postorder(root,n,0);
                break;
            case 0:
                printf("Program terminates...\n");
                exit(0);
            default:
                printf("\nInvalid Choice!!!\n");
        }
    }while(1);

    return 0;
}
```

output:

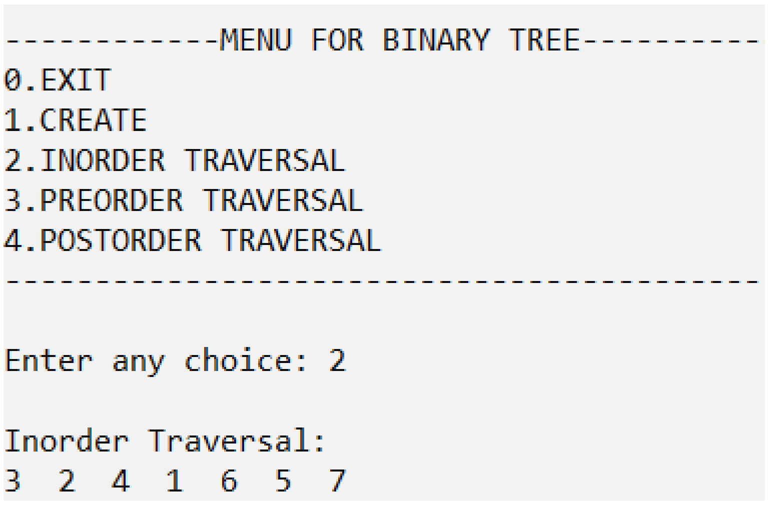


C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 4\21_04_02.exe

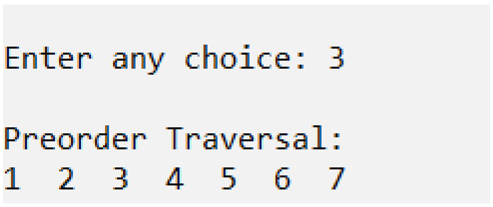
```

------------MENU FOR BINARY TREE----------------
0.EXIT
1.CREATE
2.INORDER TRAVERSAL
3.PREORDER TRAVERSAL
4.POSTORDER TRAVERSAL
------------------------------------------

Enter any choice: 1

Enter no of nodes in the binary tree: 7

Enter the data of the node: 1

Do you want to add left child of 1 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 2

Do you want to add left child of 2 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 3

Do you want to add left child of 3 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 3 (Enter 1 for yes or 0 for no) : 0
```



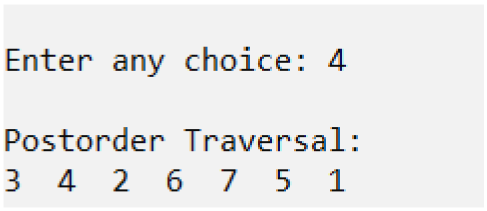
```
Do you want to add right child of 2 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 4

Do you want to add left child of 4 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 4 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 1 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 5

Do you want to add left child of 5 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 6

Do you want to add left child of 6 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 6 (Enter 1 for yes or 0 for no) : 0

Do you want to add right child of 5 (Enter 1 for yes or 0 for no) : 1

Enter the data of the node: 7

Do you want to add left child of 7 (Enter 1 for yes or 0 for no) : 1
Array is full

Do you want to add right child of 7 (Enter 1 for yes or 0 for no) : 0

Binary tree created successfully
```



```
------------MENU FOR BINARY TREE----------
0.EXIT
1.CREATE
2.INORDER TRAVERSAL
3.PREORDER TRAVERSAL
4.POSTORDER TRAVERSAL
------------------------------------------

Enter any choice: 2

Inorder Traversal:
3  2  4  1  6  5  7
```



```
Enter any choice: 3

Preorder Traversal:
1  2  3  4  5  6  7
```



```
Enter any choice: 4

Postorder Traversal:
3  4  2  6  7  5  1
```

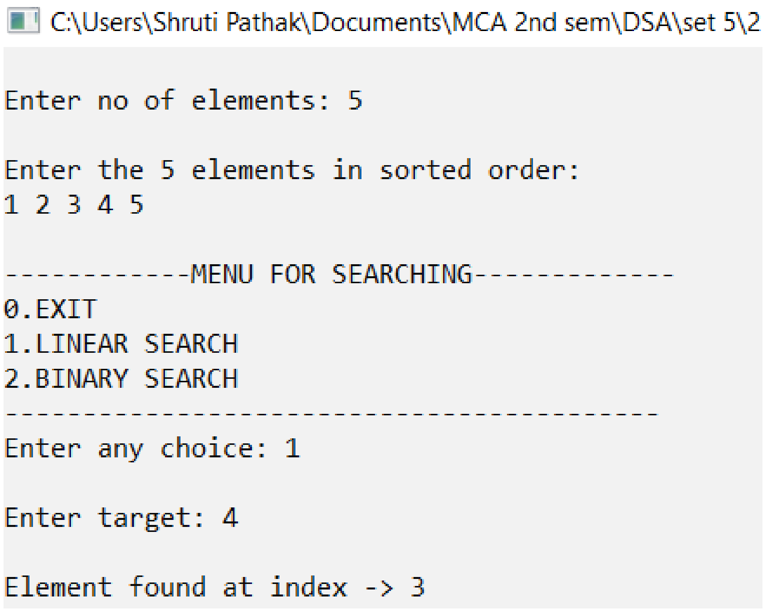
DSA ASSIGNMENT 5:

1. Write a menu-driven program to implement the following searching techniques using an array (a)Linear or sequential search (b) Binary search

SOURCE CODE:

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

int linearSearch(int *a, int s, int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (a[i] == s)
        {
            return i;
        }
    }
    return -1;
}
int binarySearch(int *a, int s, int n)
{
    int l=0,r=n-1;
    while(l<=r)
    {
        int m=(l+r)/2;
        if(a[m]<s)
            l=m+1;
        else if(a[m]>s)
            r=m-1;
        else
        {
            return m;
        }
    }
    return -1;
}
int main()
{
    int *a,n,t,ch,p,i;
    printf("\nEnter no of elements: ");
    scanf("%d",&n);
    a=(int*)malloc(sizeof(int)*n);
    printf("\nEnter the %d elements in sorted order:\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    do
    {
        printf("\n------------MENU FOR SEARCHING-------------\n");
        printf("0.EXIT\n");
        printf("1.LINEAR SEARCH\n");
        printf("2.BINARY SEARCH\n");
        printf("------------------------------------------\n");

        printf("Enter any choice: ");
        scanf("%d",&ch);
        switch (ch)
        {
            case 1:
                printf("\nEnter target: ");
                scanf("%d",&t);
                p=linearSearch(a,t,n);
                if(p>=0)
                    printf("\nElement found at index -> %d",p);
                else
                    printf("\nElement not found");
                break;
            case 2:
                printf("\nEnter target: ");
                scanf("%d",&t);
                p=binarySearch(a,t,n);
                if(p>=0)
                    printf("\nElement found at index -> %d",p);
                else
                    printf("\nElement not found");
                break;
            case 0:
                exit(0);
            default:
            printf("\nInvalid input!");
            break;
        }
    } while (1);
    return 0;
}
```

output:



C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 5\2

```
Enter no of elements: 5

Enter the 5 elements in sorted order:
1 2 3 4 5

------------MENU FOR SEARCHING-------------
0.EXIT
1.LINEAR SEARCH
2.BINARY SEARCH
------------------------------------------
Enter any choice: 1

Enter target: 4

Element found at index -> 3
```

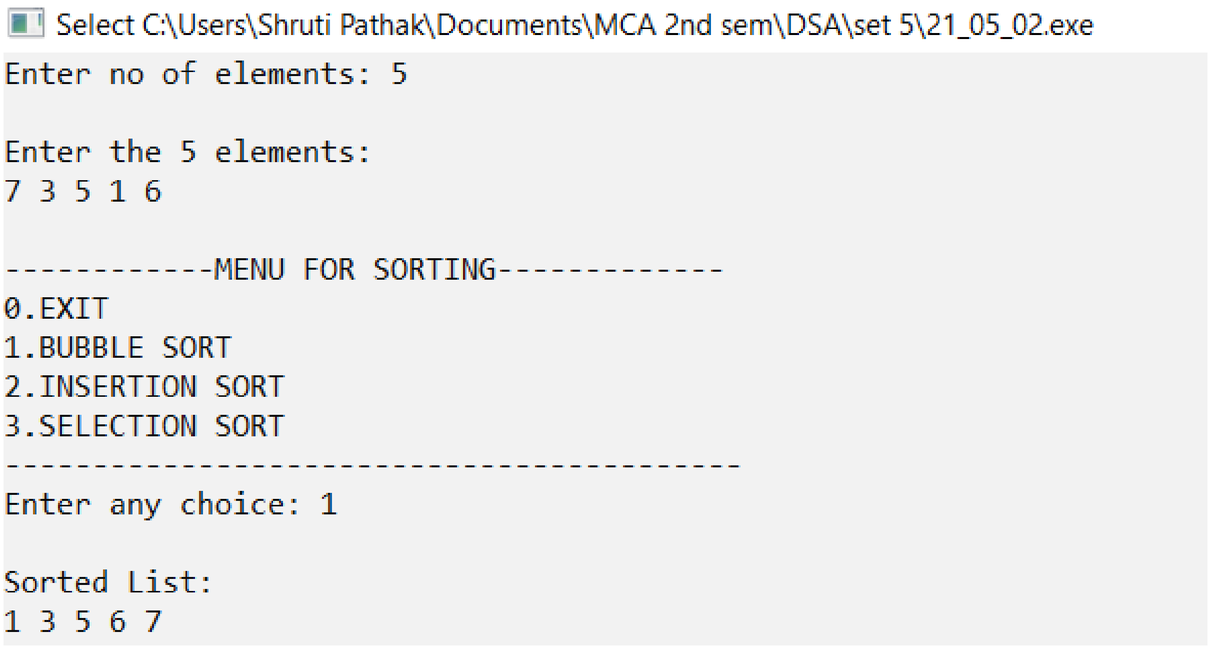
```
-----------------------------------
Enter any choice: 2

Enter target: 3

Element found at index -> 2
```

2. Write a menu-driven program to implement the following sorting techniques using an array (a) Bubble sort (b) Insertion sort (c) Selection sort

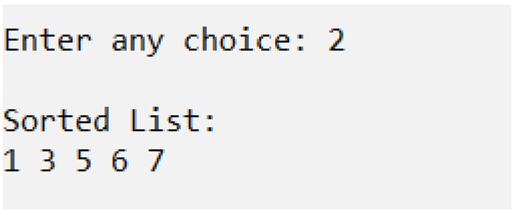
SOURCE CODE:

```c
#include <stdio.h>
#include <stdlib.h>
void bubbleSort(int *a, int n)
{
    int i,j;
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                int t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
}
void insertSort(int *a, int n)
{
     int i, j, key;

    for (i = 1; i < n; i++) {
        key = a[i];
        j = i - 1;

        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
            j = j - 1;
        }

        a[j + 1] = key;
    }
}
void selectionSort(int *a,int n)
{
    int i, j, minIndex, temp;

    for (i = 0; i < n - 1; i++) {
        minIndex = i;

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

        temp = a[i];
        a[i] = a[minIndex];
        a[minIndex] = temp;
    }
}

void displayArray(int *a, int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

int main()
{
    int ch,*a,n,i;
    printf("Enter no of elements: ");
    scanf("%d",&n);
    a=(int*)malloc(sizeof(int)*n);
    printf("\nEnter the %d elements:\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    do
    {
        printf("\n------------MENU FOR SORTING-------------\n");
        printf("0.EXIT\n");
        printf("1.BUBBLE SORT\n");
        printf("2.INSERTION SORT\n");
        printf("3.SELECTION SORT\n");
        printf("------------------------------------------\n");

        printf("Enter any choice: ");
        scanf("%d",&ch);
        switch (ch)
        {
            case 0:
                exit(0);
            case 1:
                bubbleSort(a,n);
                printf("\nSorted List:\n");
                displayArray(a,n);
                break;
            case 2:
                insertSort(a,n);
                printf("\nSorted List:\n");
                displayArray(a,n);
                break;
            case 3:
                selectionSort(a,n);
                printf("\nSorted List:\n");
                displayArray(a,n);
                break;
            default:
                printf("\nInvalid input!");
                break;
            }
    } while (1);
    return 0;
}
```

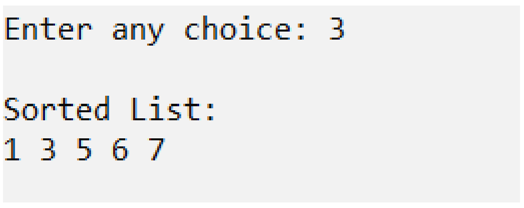
output:



```
Select C:\Users\Shruti Pathak\Documents\MCA 2nd sem\DSA\set 5\21_05_02.exe
Enter no of elements: 5

Enter the 5 elements:
7 3 5 1 6

------------MENU FOR SORTING-------------
0.EXIT
1.BUBBLE SORT
2.INSERTION SORT
3.SELECTION SORT
------------------------------------------
Enter any choice: 1

Sorted List:
1 3 5 6 7
```



```
Enter any choice: 2

Sorted List:
1 3 5 6 7
```



```
Enter any choice: 3

Sorted List:
1 3 5 6 7
```