Name: Jain Apurva Sanjay

Roll no: 12

Class: MCA Sem1

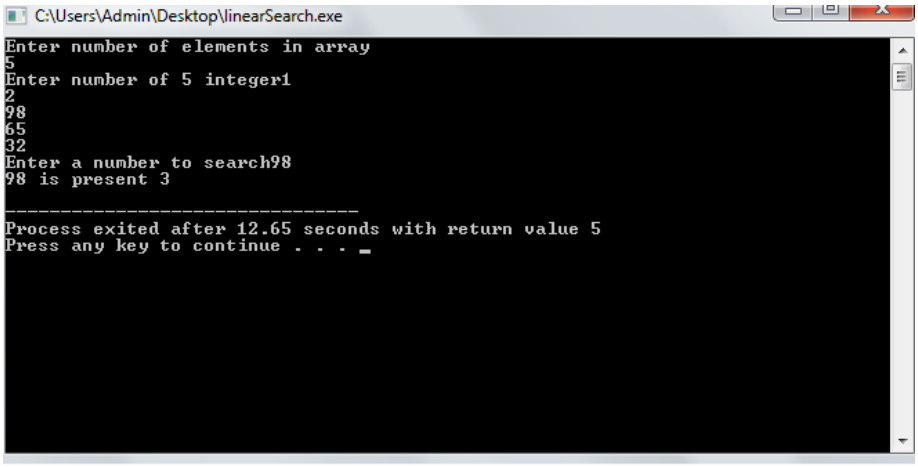
Div: A

1. **Write a program implement Linear Search.**

Code:

```
#include<stdio.h> int main()
{
int a[100],search,i,n;
printf("Enter number of elements in array\n"); scanf("%d",&n);
printf("Enter number of %d integer",n); for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter a number to search"); scanf("%d",&search); for(i=0;i<n;i++)
if(a[i]==search)
{
        printf("%d is present %d\n",search,i+1); break;
}
if(i==n)
printf("%d is present %d\n",search,i+1); break;

printf("%d is not present",search);
}
```

Output:



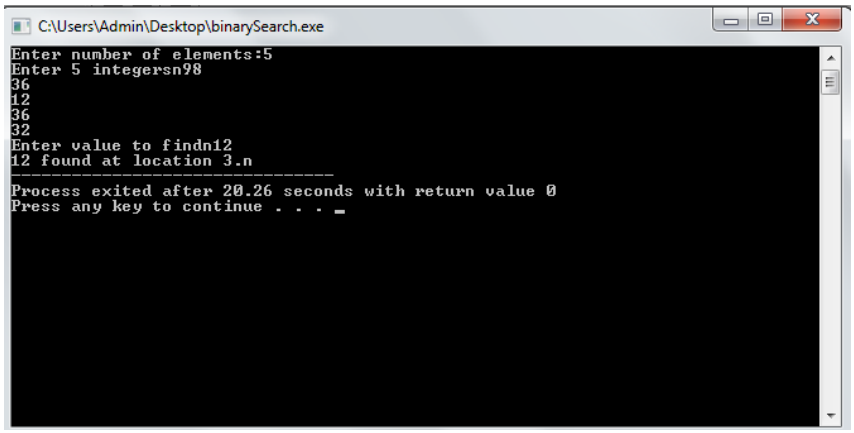
2. **Write a program to implement Binary Search.**

Code:

```
#include <stdio.h>

int main()

{

int i, low, high, mid, n, key, array[100]; printf("Enter number of elements:"); scanf("%d",&n);

printf("Enter %d integers", n); for(i = 0; i < n; i++) scanf("%d",&array[i]); printf("Enter value to find"); scanf("%d",
&key);

low = 0; high = n - 1;

mid = (low+high)/2; while (low <= high)

{

if(array[mid] < key)

{

low = mid + 1;

}

else if (array[mid] == key)

{

printf("%d found at location %d.n", key, mid+1); break;

}

else
```

```
high = mid - 1;
mid = (low + high)/2;
}
if(low > high)
printf("Not found! %d isn't present in the list.n", key); return 0;
```

Output:



3. **Write a menu driven program to implement the following String Operations.**

   uppercase 2) strcat 3) lowercase 4) strcpy 5) strlen 6) strcmp

Code:

```
#include <stdio.h>
void myUppercase() {
   char str[100];
   printf("Enter a string: ");
   gets(str);
   for (int i = 0; str[i]; i++) {
      if (str[i] >= 'a' && str[i] <= 'z') {
         str[i] = str[i] - 32;
      }
   }
```

```
    printf("Uppercase string: %s\n", str);
}
void myLowercase() {
    char str[100];
    printf("Enter a string: ");
    gets(str);

    for (int i = 0; str[i]; i++) {
        if (str[i] >= 'A' && str[i] <= 'Z') {
            str[i] = str[i] + 32;
        }
    }
    printf("Lowercase string: %s\n", str);
}
int myStringLength() {
    char str[100];
    printf("Enter a string: ");
    gets(str);
    int length = 0;
    for (int i = 0; str[i]; i++) {
        length++;
    }
    printf("Length of the string: %d\n", length);
    return length;
}
void myStringConcatenation() {
    char str1[100], str2[100];
    printf("Enter the first string: ");
    gets(str1);
```

```
    printf("Enter the second string: ");
    gets(str2);
    int i = 0, j = 0;
    while (str1[i] != '\0') {
        i++;
    }
    while (str2[j] != '\0') {
        str1[i] = str2[j];
        i++;
        j++;
    }
    str1[i] = '\0';
    printf("Concatenated string: %s\n", str1);
}
void myStringCopy() {
    char str1[100], str2[100];
    printf("Enter a string to copy: ");
    gets(str1);
    int i = 0;
    while (str1[i] != '\0') {
        str2[i] = str1[i];
        i++;
    }
    str2[i] = '\0';
    printf("Copied string: %s\n", str2);
}
int myStringComparison() {
    char str1[100], str2[100];
    printf("Enter the first string: ");
```

```
    gets(str1);
    printf("Enter the second string: ");
    gets(str2);
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0') {
        if (str1[i] != str2[i]) {
            printf("Strings are not equal.\n");
            return -1;
        }
        i++;
    }
    if (str1[i] == '\0' && str2[i] == '\0') {
        printf("Strings are equal.\n");
        return 0;
    } else {
        printf("Strings are not equal.\n");
        return 1;
    }
}
int main() {
    int choice;
    while (1) {
        printf("\nString Operations Menu:\n");
        printf("1. Uppercase\n");
        printf("2. Lowercase\n");
        printf("3. String Length\n");
        printf("4. String Concatenation\n");
        printf("5. String Copy\n");
        printf("6. String Comparison\n");
```

```
printf("7. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
getchar(); // Consume the newline character left in the input buffer
switch (choice) {
    case 1:
        myUppercase();
        break;
    case 2:
        myLowercase();
        break;
    case 3:
        myStringLength();
        break;
    case 4:
        myStringConcatenation();
        break;
    case 5:
        myStringCopy();
        break;
    case 6:
        myStringComparison();
        break;
    case 7:
        printf("Exiting the program. Goodbye!\n");
        return 0;
    default:
        printf("Invalid choice. Please enter a valid option.\n");
}
```

```
    }
    return 0;
}
```

Output:

```
String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 1
Enter a string: apurva
Uppercase string: APURVA

String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 2
Enter a string: APURVA
Lowercase string: apurva

String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 3
Enter a string: APURVA
Length of the string: 6
```

```
String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 4
Enter the first string: APUR
Enter the second string: JAI
Concatenated string: APURVAJ

String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 5
Enter a string to copy: APUR
Copied string: APURVA

String Operations Menu:
1. Uppercase
2. Lowercase
3. String Length
4. String Concatenation
5. String Copy
6. String Comparison
7. Exit
Enter your choice: 6
Enter the first string: APUR
```

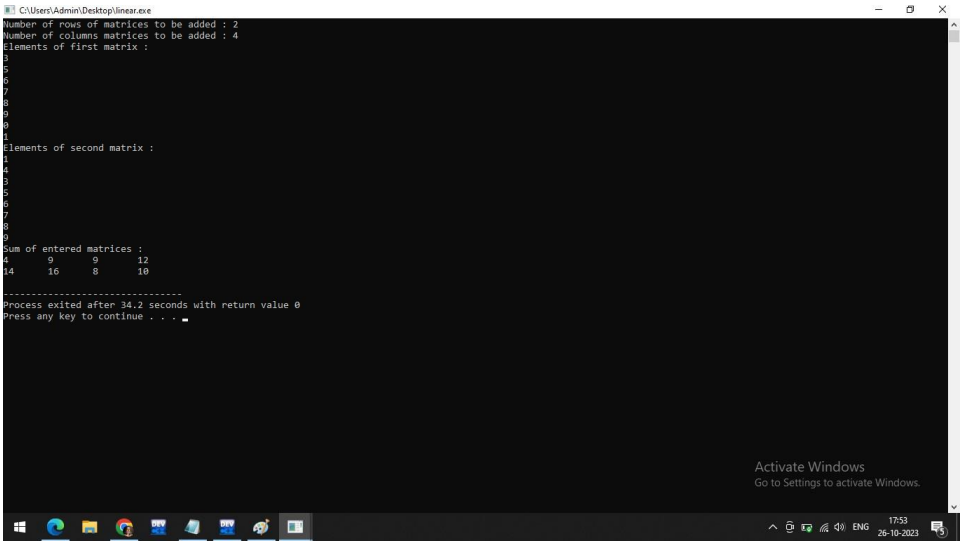
4. **Write a program which do addition of 2 matrix.**

Code:

```
#include <stdio.h>

int main()

{
```

```
int rowCount, columnCount, i, j;
int firstMatrix[10][10], secondMatrix[10][10], resultMatrix[10][10];
printf("Number of rows of matrices to be added : ");scanf("%d", &rowCount);
printf("Number of columns matrices to be added : ");scanf("%d", &columnCount);
printf("Elements of first matrix : \n");for (i = 0; i < rowCount; i++)
for (j = 0; j < columnCount; j++)
scanf("%d", &firstMatrix[i][j]); printf("Elements of second matrix : \n");
for (i = 0; i < rowCount; i++)
for (j = 0; j < columnCount; j++) scanf("%d", &secondMatrix[i][j]);
printf("Sum of entered matrices : \n")
for (i = 0; i < rowCount; i++)
{
for (j = 0; j < columnCount; j++)
{
resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];printf("%d\t",resultMatrix[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:


C:\Users\Admin\Desktop\linear.exe
Number of rows of matrices to be added : 2
Number of columns matrices to be added : 4
Elements of first matrix :
1
2
3
4
5
6
7
8
9
0
1
Elements of second matrix :
1
2
3
4
5
6
7
8
9
0
Sum of entered matrices :
4 9 9 12
14 16 8 10

Process exited after 34.2 seconds with return value 0
Press any key to continue . . .
Activate Windows
Go to Settings to activate Windows.
ENG
17:53
26-10-2023

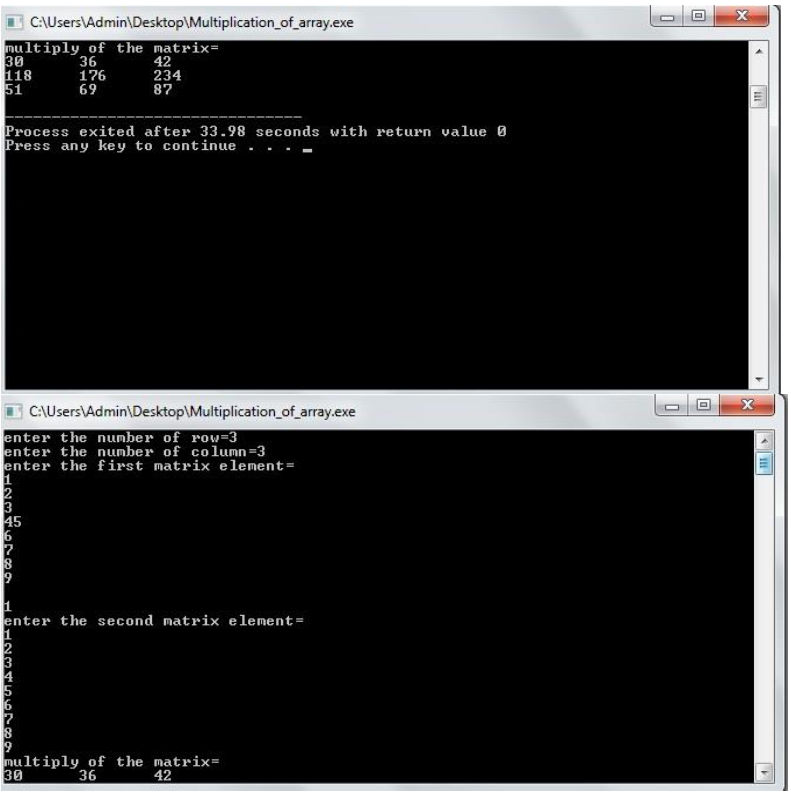
5. **Write a program which do multiplication of 2 matrix.**

Code:

```
#include<stdio.h>
 #include<stdlib.h>
int main()
{
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
system("cls");
printf("enter the number of row=");scanf("%d",&r);
printf("enter the number of column=");scanf("%d",&c);
printf("enter the first matrix element=\n");for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element=\n");for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("multiply of the matrix=\n");for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0; for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
```

```
}
}
}
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:

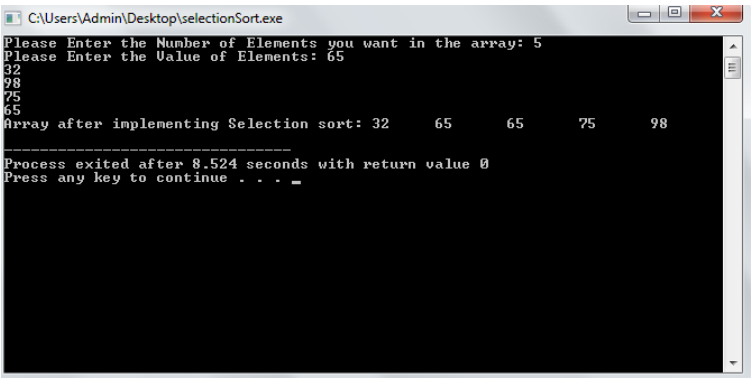


6. **Write a program to implement Selection Sort.**

Code:

```
#include <stdio.h>

int main()

{

int arr[50],n;

int i, j, position, swap;

printf("Please Enter the Number of Elements you want in the array: ");scanf("%d",&n);

printf("Please Enter the Value of Elements: ");

for(i=0;i<n;i++) scanf("%d",&arr[i]);

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

{

position = i;

for (j = i + 1; j < n; j++)

{

if (arr[position] > arr[j])position = j;

}

if (position != i)

{

swap = arr[i];

arr[i] = arr[position];arr[position] = swap;

}

}

printf("Array after implementing Selection sort: ");for (i = 0; i < n; i++)

printf("%d\t", arr[i]);return 0;

}
```

Output:



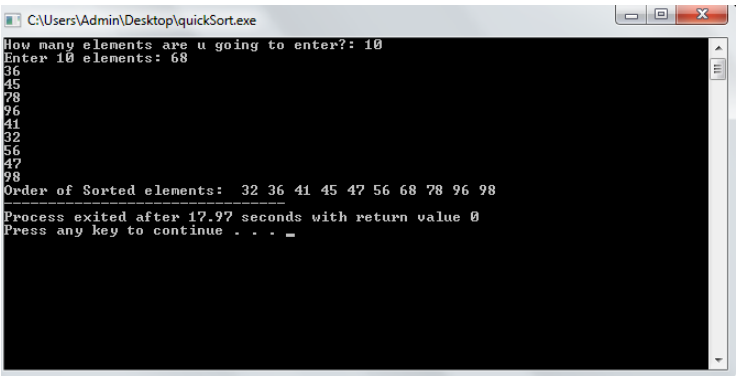
7. **Write a program to implement Bubble Sort.**

Code:

```
#include <stdio.h>

int main()

{

int a[50], num, i, j, temp;

printf("Please Enter the Number of Elements you want in the array: ");scanf("%d", &num);

printf("Please Enter the Value of Elements: ");for(i = 0; i < num; i++)

scanf("%d", &a[i]);

for(i = 0; i < num - 1; i++)

{

for(j = 0; j < num - i - 1; j++)

{

if(a[j] > a[j+ 1])

{

temp =  a[j]; a[j] = a[j + 1]; a[j + 1] = temp;

}

}

}
```

```
printf("Array after implementing bubble sort: ");for(i = 0; i < num; i++){

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

}

return 0;

}
```

Output:



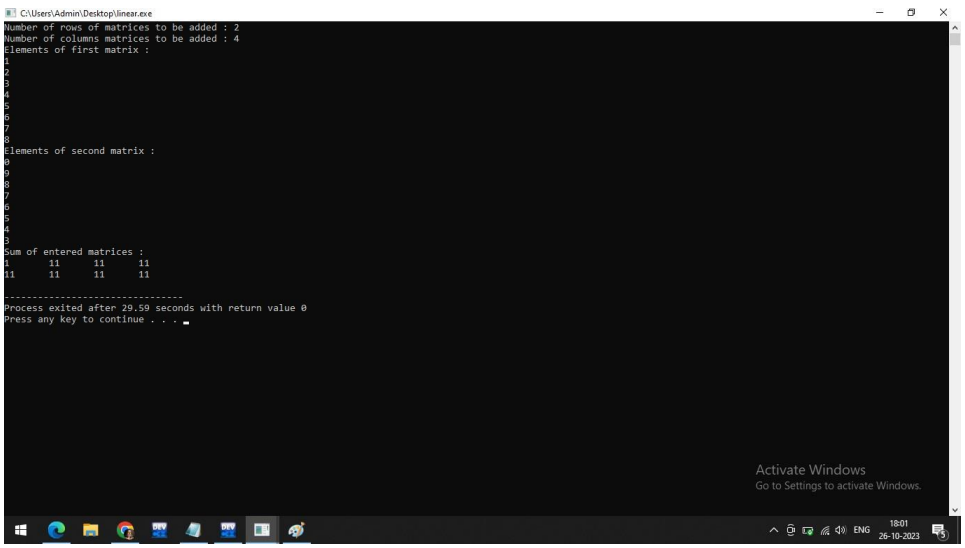
8. Write a program to implement Quick Sort.

Code:

```
#include<stdio.h>

void quicksort(int number[25],int first,int last)

{

int i, j, pivot, temp;if(first<last){

pivot=first;i=first; j=last; while(i<j){

while(number[i]<=number[pivot]&&i<last)i++;

while(number[j]>number[pivot])j--;

if(i<j){ temp=number[i]; number[i]=number[j];number[j]=temp;

}

}

temp=number[pivot]; number[pivot]=number[j]; number[j]=temp; quicksort(number,first,j-1);
quicksort(number,j+1,last);

}

}
```

```
int main()

{

int i, count, number[25];

printf("How many elements are u going to enter?: ");scanf("%d",&count);

printf("Enter %d elements: ", count);for(i=0;i<count;i++) scanf("%d",&number[i]); quicksort(number,0,count-1);
printf("Order of Sorted elements: ");for(i=0;i<count;i++)

printf(" %d",number[i]);return 0;

}
```

Output:



9. Write a program to implement Stack Operation.

Code:

```
#include<stdio.h>

 #include<conio.h>

#define N 5

int stack[N];int top =-1; int i,j;

void push(); void pop(); void dis(); void update();int main(){

int ch;do

{

printf("\n");
```

```
printf("1.FOR PUSH \n"); printf("2.FOR POP \n"); printf("3.FOR DIS \n"); printf("4.FOR Update \n");printf("5.FOR
Exit \n"); scanf("%d",&ch);

switch (ch)

{

case 1:

push();break;

case 2:

pop();break;

case 3:

dis(); break;

case 4:

update();break;

default:

break;

}

} while (ch!=5);

return 0;

}

void push(){ if (top==N-1)

{

printf("Stack is full");

}

else{

int val;

printf("Enter the value :");scanf("%d",&val);

top++; stack[top] = val;

}

}

void dis(){ printf("\n");

for (int i = 0; i <=top; i++)

{
```

```
printf(" %d ",stack[i]);
}
}
void pop(){
if (top==-1)
{
printf("Stackk is EMpty");
}
else{
int popedel = stack[top];top--;
printf("POPED ELEMENT IS : %d", popedel);dis();
}
}
void update()
{
int idx,newval; printf("Enter the Index :");scanf("%d",&idx);
if (idx<0 || idx>top)
{
printf("INVALID INDEX \n");
}
else{
printf("Enter the value for Update :");scanf("%d",&newval);
stack[idx] = newval;
printf(" UPDATED VALUE %d at INDEX %d",newval,idx);dis();
}
}
```

Output:

```
1.FOR PUSH
2.FOR POP
3.FOR DIS
4.FOR Update
5.FOR Exit
1
Enter the value :2
1.FOR PUSH
2.FOR POP
3.FOR DIS
4.FOR Update
5.FOR Exit
1
Enter the value :3
1.FOR PUSH
2.FOR POP
3.FOR DIS
4.FOR Update
5.FOR Exit
3
2  3
1.FOR PUSH
```

10. Write a program to implement Infix to Postfix Operation.

Code:

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

#define MAX_SIZE 100

struct Stack {

char items[MAX_SIZE]; int top;

};

void initialize(struct Stack* stack) { stack->top = -1;

}

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

}

void push(struct Stack* stack, char item) { stack->items[++stack->top] = item;

}

char pop(struct Stack* stack) { if (!isEmpty(stack)) {

return stack->items[stack->top--];

}
```

```
return '\0'; // Null character represents an empty stack
}
char peek(struct Stack* stack) {
if (!isEmpty(stack)) {
return stack->items[stack->top];
}
return '\0'; // Null character represents an empty stack
}
int isOperator(char c) {
return (c == '+' || c == '-' || c == '*' || c == '/');
}
int getPrecedence(char c) { if (c == '+' || c == '-')
return 1;
if (c == '*' || c == '/') return 2;
return 0;
}
void infixToPostfix(const char* infix, char* postfix) { struct Stack operatorStack; initialize(&operatorStack);
int postfixIndex = 0;
for (int i = 0; infix[i]; i++) { char currentChar = infix[i];
if (isalnum(currentChar)) { postfix[postfixIndex++] = currentChar;
} else if (currentChar == '(') { push(&operatorStack, currentChar);
} else if (currentChar == ')') {
while (!isEmpty(&operatorStack) && peek(&operatorStack) != '(') {
postfix[postfixIndex++] = pop(&operatorStack);
}
if (!isEmpty(&operatorStack) && peek(&operatorStack) != '(') { printf("Invalid expression\n");
return;
} else {
pop(&operatorStack); // Pop the opening parenthesis
}
} else if (isOperator(currentChar)) {
```

```
while (!isEmpty(&operatorStack) && getPrecedence(currentChar) <= getPrecedence(peek(&operatorStack))) {
postfix[postfixIndex++] = pop(&operatorStack);
}
push(&operatorStack, currentChar);
}
}
while (!isEmpty(&operatorStack)) {
char topOperator = pop(&operatorStack); if (topOperator == '(') {
printf("Invalid expression\n"); return;
}
postfix[postfixIndex++] = topOperator;
}
postfix[postfixIndex] = '\0';
}

int main() {
char infix[MAX_SIZE]; char postfix[MAX_SIZE];
printf("Enter an infix expression: "); fgets(infix, sizeof(infix), stdin);
infixToPostfix(infix, postfix);
printf("Postfix Expression: %s\n", postfix);
return 0;
}
```

Output:

```
/tmp/CAgqXWwhqz.o
Enter an infix expression: $#%123
Postfix Expression: 123
```

[Type here]