**1. Menu-driven C program to swap the values of 2 variables using and without using temporary**

**variable.**

procedure swap(a, b)

set temp to 0

temp ← a

a ← b

b ← temp

end procedure

#include<stdio.h>

int main()

{

double first, second, temp;

printf("Enter first number: ");

scanf("%lf", &first);

printf("Enter second number: ");

scanf("%lf", &second);

temp = first;

first = second;

second = temp;

printf("\nAfter swapping, firstNumber = %.2lf\n", first);

printf("After swapping, secondNumber = %.2lf", second);

return 0;

}



#include <stdio.h>

int main() {

double a, b;

printf("Enter a: ");

scanf("%lf", &a);

printf("Enter b: ");

scanf("%lf", &b);

a = a - b;

b = a + b;

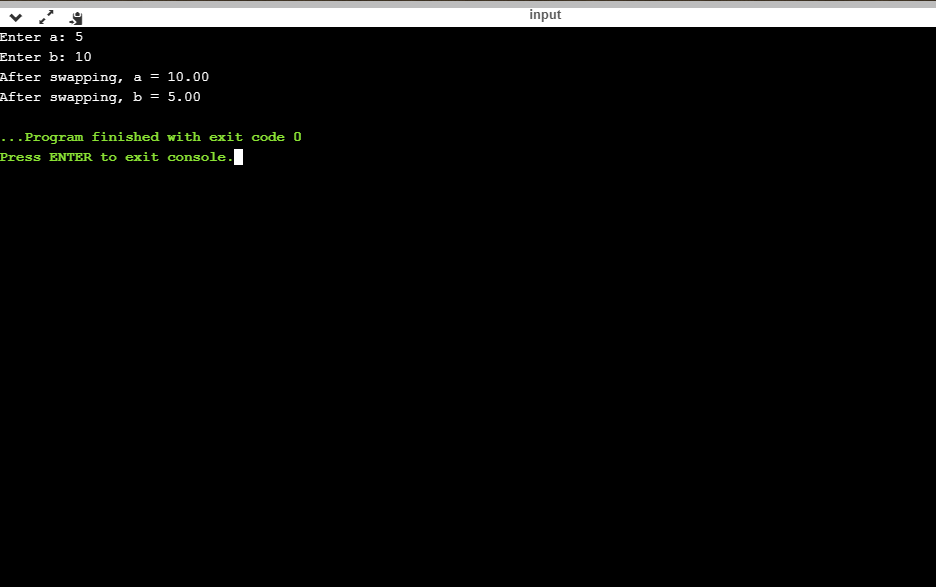
a = b - a;

printf("After swapping, a = %.2lf\n", a);

printf("After swapping, b = %.2lf", b);

return 0;

}



**2. Menu-driven C program to swap the values of 3 variables using and without using temporary**

**variable.**

procedure swap(a, b)

set temp to 0

temp ← a

a ← b

b ← c

c ← temp

end procedure

#include <stdio.h>

int main() {

double a, b, c, d;

printf("Enter a: ");

scanf("%lf", &a);

printf("Enter b: ");

scanf("%lf", &b);

printf("Enter c: ");

scanf("%lf", &c);

d = a;

a = b;

b = c;

c = d;

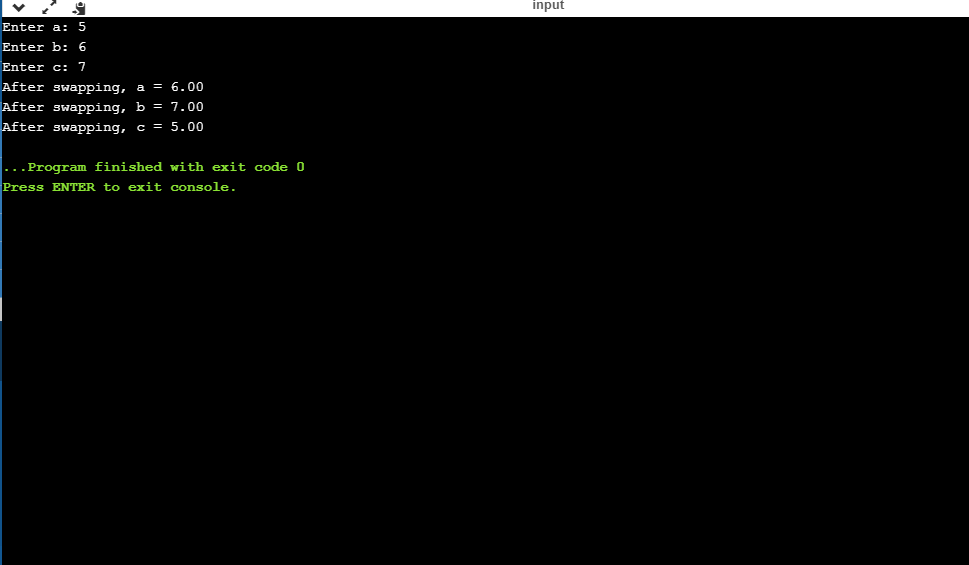
printf("After swapping, a = %.2lf\n", a);

printf("After swapping, b = %.2lf\n", b);

printf("After swapping, c = %.2lf", c);

return 0;

}



#include <stdio.h>

int main() {

double a, b, c;

printf("Enter a: ");

scanf("%lf", &a);

printf("Enter b: ");

scanf("%lf", &b);

printf("Enter c: ");

scanf("%lf", &c);

a = a+b+c;

b = a-b-c;

c = a-b-c;

a = a-b-c

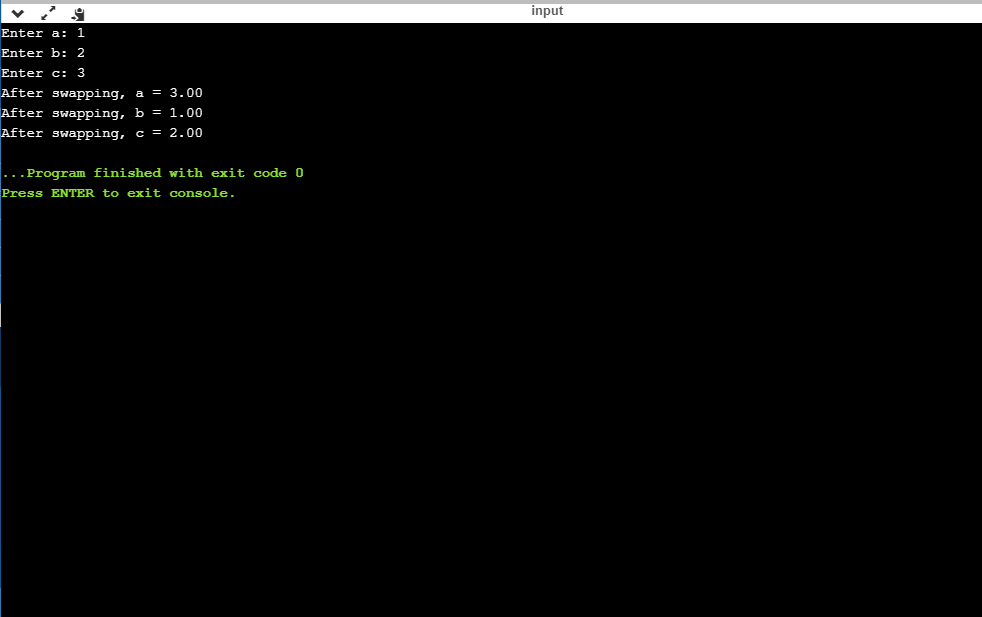
printf("After swapping, a = %.2lf\n", a);

printf("After swapping, b = %.2lf", b);

printf("After swapping, c = %.2lf", c);

return 0;

}



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

SET Max to array[0]

FOR i = 1 to array length - 1

IF array[i] > Max THEN

SET Max to array[i]

ENDIF

ENDFOR

PRINT Max

#include <stdio.h>

#include<stdlib.h>

int i, max, min, size, arr[50];

void maximum();

void minimum();

int main()

{

int choice;

do

{

printf("\n\n\----------MENU----------\n");

printf("1. MAX\n");

printf("2. MIN\n");

printf("-------------------------");

printf("\nPlease enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

maximum();

break;

case 2:

minimum();

break;

default:

printf("\nYou have entered an invalid choice");

break;

}

}while(choice!=3);

return 0;

}

void maximum()

{

printf("Enter size of the array: ");

scanf("%d", &size);

printf("Enter elements in the array: ");

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

{

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

}

max = arr[0];

min = arr[0];

for(i=1; i<size; i++)

{

if(arr[i] > max)

{

max = arr[i];

}

}

printf("Maximum element = %d\n", max);

}

void minimum()

{

printf("Enter size of the array: ");

scanf("%d", &size);

printf("Enter elements in the array: ");

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

{

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

}

min = arr[0];

for(i=1; i<size; i++)

{

if(arr[i] < min)

{

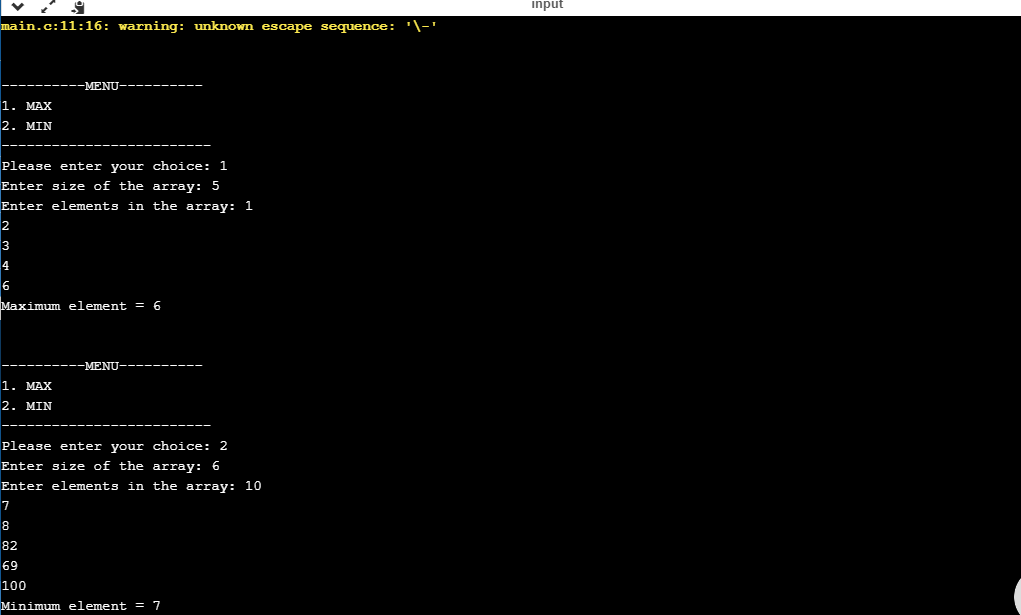
min = arr[i];

}

}

printf("Minimum element = %d", min);

}



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

SET Lo to 0

SET Hi to array length - 1

WHILE Lo <= Hi

SET Mid to (Lo + Hi) / 2

IF X < array[Mid] THEN

SET Hi to Mid - 1

ELSE IF X > array[Mid] THEN

SET Lo to Mid + 1

ELSE

RETURN Mid

ENDIF

ENDWHILE

RETURN -1

#include<stdio.h>

#include<stdlib.h>

int c, first, last, middle, n, search, array[100];

void binarySearch();

void linearSearch();

int main()

{

int choice;

do

{

printf("\n\n\----------MENU----------\n");

printf("1. Binary search\n");

printf("2. Linear search\n");

printf("3. Exit\n");

printf("\nPlease enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

binarySearch();

break;

case 2:

linearSearch();

break;

case 3:

exit(0);

break;

default:

printf("\nYou have entered an invalid choice");

break;

}

}while(choice!=4);

return 0;

}

void binarySearch()

{

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last) {

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search) {

printf("%d found at location %d.\n", search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found! %d isn't present in the list.\n", search);

return 0;

}

void linearSearch()

{

printf("Enter number of elements in array\n");

scanf("%d", &n);

printf("Enter %d integer(s)\n", n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

printf("Enter a number to search\n");

scanf("%d", &search);

for (c = 0; c < n; c++)

{

if (array[c] == search) /\* If required element is found \*/

{

printf("%d is present at location %d.\n", search, c+1);

break;

}

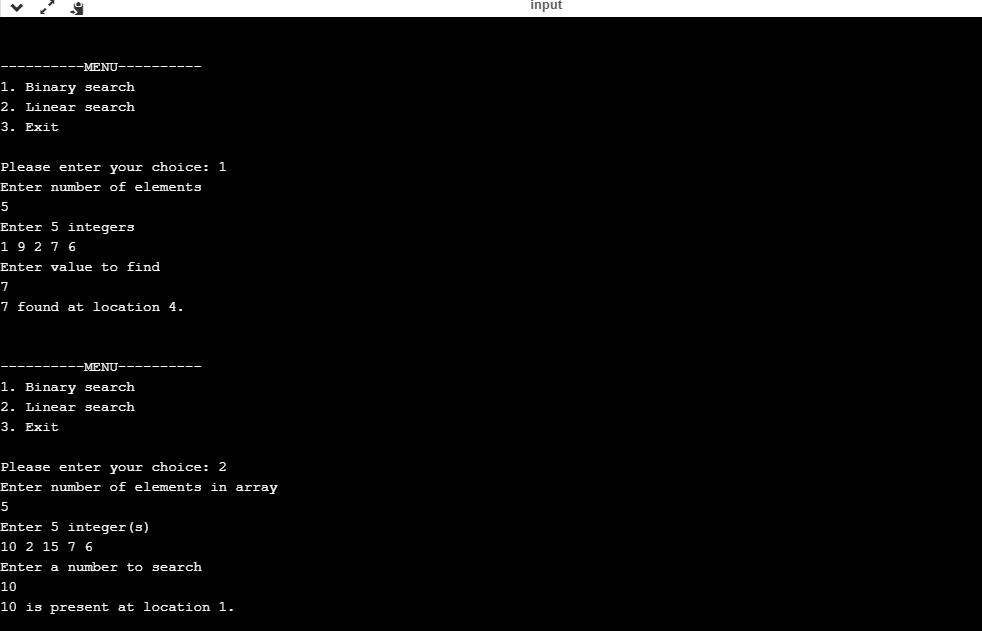
}

if (c == n)

printf("%d isn't present in the array.\n", search);

return 0;

}



**5. Menu-driven C program to perform insertion, deletion, display and search operations in an**

**ordered array.**

#include<stdio.h>

#include<stdlib.h>

int a[50],b[50],c[50];

int m,n,p,pos,key,i,val;

void create();

void insert();

void display();

void del();

void search();

int main()

{

int choice;

do

{

printf("\n\n\----------MENU----------\n");

printf("1. Create\n");

printf("2. Insert\n");

printf("3. Display\n");

printf("4. Delete\n");

printf("5. Search\n");

printf("6. Exit\n");

printf("-------------------------");

printf("\nPlease enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

insert();

break;

case 3:

display();

break;

case 4:

del();

break;

case 5:

search();

break;

case 6:

exit(0);

break;

default:

printf("\nYou have entered an invalid choice");

break;

}

}while(choice!=7);

return 0;

}

void create()

{

printf("\nPlease enter the size of the array: ");

scanf("%d",&n);

printf("\nEnter the ordered array: ");

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

{

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

}

}

void display()

{

int i;

printf("\nThe elements of the ordered array are: ");

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

{

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

}

}

void insert()

{

printf("\nEnter the position of the new element: ");

scanf("%d",&pos);

printf("\nEnter the value of the new element: ");

scanf("%d",&val);

for(i=n-1;i>=1;i--)

{

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

}

a[pos]=val;

n=n+1;

}

void del()

{

printf("\nEnter the position o the element to be deleted: ");

scanf("%d",&pos);

val=a[pos];

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

{

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

}

n=n-1;

printf("\nThe deleted element from the array is = %d",val);

}

void search()

{

printf("\nEnter the element to be searched for: ");

scanf("%d",&key);

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

{

if(a[i]==key)

{

printf("\nThe element is present at the position = %d",i);

break;

}

if(i==n)

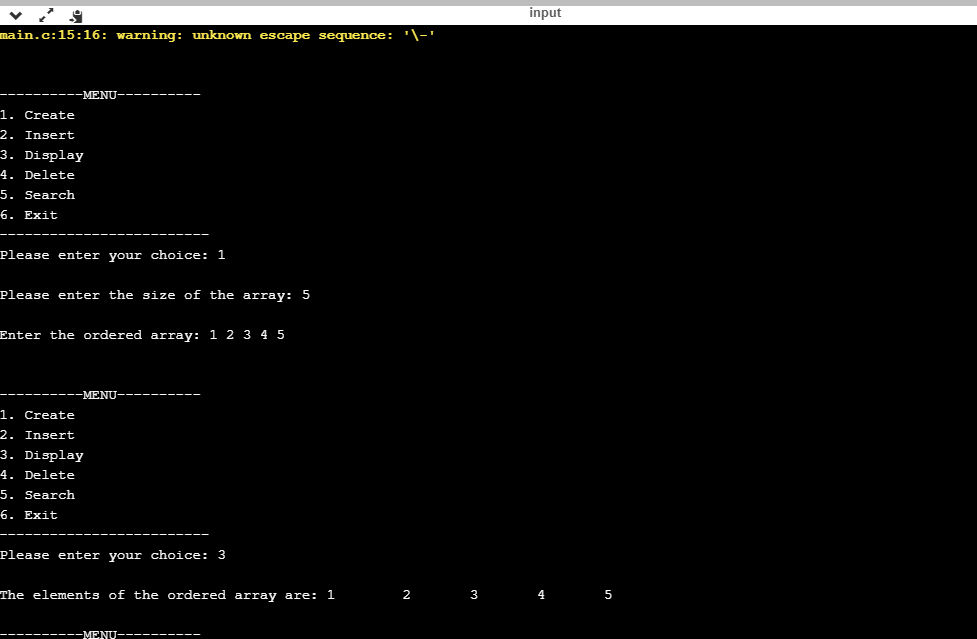
{

printf("\nThe search was unsuccessful");

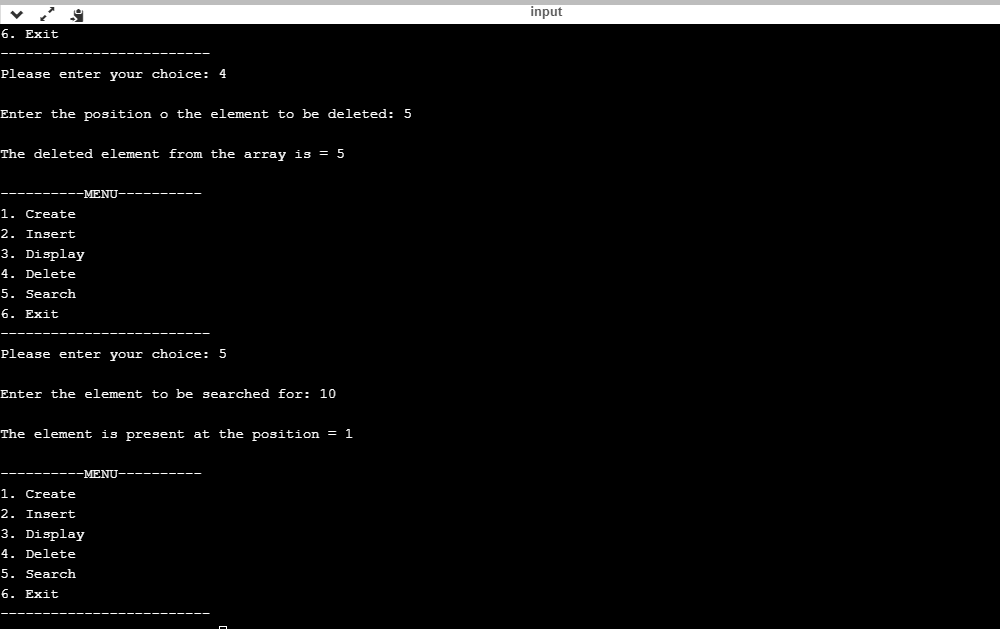
}

}

}







**6. Menu-driven C program to perform insertion, deletion, display and search operations in an**

**unordered array.**

#include<stdio.h>

#include<stdlib.h>

int a[50],b[50],c[50];

int m,n,p,pos,key,i,val;

void create();

void insert();

void display();

void del();

void search();

int main()

{

int choice;

do

{

printf("\n\n\----------MENU----------\n");

printf("1. Create\n");

printf("2. Insert\n");

printf("3. Display\n");

printf("4. Delete\n");

printf("5. Search\n");

printf("6. Exit\n");

printf("-------------------------");

printf("\nPlease enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

insert();

break;

case 3:

display();

break;

case 4:

del();

break;

case 5:

search();

break;

case 6:

exit(0);

break;

default:

printf("\nYou have entered an invalid choice");

break;

}

}while(choice!=7);

return 0;

}

void create()

{

printf("\nPlease enter the size of the array: ");

scanf("%d",&n);

printf("\nEnter the ordered array: ");

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

{

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

}

}

void display()

{

int i;

printf("\nThe elements of the ordered array are: ");

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

{

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

}

}

void insert()

{

printf("\nEnter the position of the new element: ");

scanf("%d",&pos);

printf("\nEnter the value of the new element: ");

scanf("%d",&val);

for(i=n-1;i>=1;i--)

{

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

}

a[pos]=val;

n=n+1;

}

void del()

{

printf("\nEnter the position o the element to be deleted: ");

scanf("%d",&pos);

val=a[pos];

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

{

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

}

n=n-1;

printf("\nThe deleted element from the array is = %d",val);

}

void search()

{

printf("\nEnter the element to be searched for: ");

scanf("%d",&key);

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

{

if(a[i]==key)

{

printf("\nThe element is present at the position = %d",i);

break;

}

if(i==n)

{

printf("\nThe search was unsuccessful");

}

}

}

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

**Push(S,x)**

**if Stack-Full(S)**

**then error "overflow"**

**else top(S) = top(S) + 1**

**S[top(S)] = x**

**Pop(S)**

**if Stack-Empty(S)**

**then error "underflow"**

**else top(S) = top(S) - 1**

**return S[top(S) + 1]**

**Stack-Empty(S)**

**if top(S) = 0**

**then return True**

**else return False**

**Stack-Full(S)**

**if top(S) = length(S)**

**then return True**

**else return False**

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int top=-1,stack[MAX];

void push();

void pop();

void display();

void main()

{

int ch;

while(1)

{

printf("\n\*\*\* Stack Menu \*\*\*");

printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");

printf("\n\nEnter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: push();

break;

case 2: pop();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nWrong Choice!!");

}

}

}

void push()

{

int val;

if(top==MAX-1)

{

printf("\nStack is full!!");

}

else

{

printf("\nEnter element to push:");

scanf("%d",&val);

top=top+1;

stack[top]=val;

}

}

void pop()

{

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

printf("\nDeleted element is %d",stack[top]);

top=top-1;

}

}

void display()

{

int i;

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

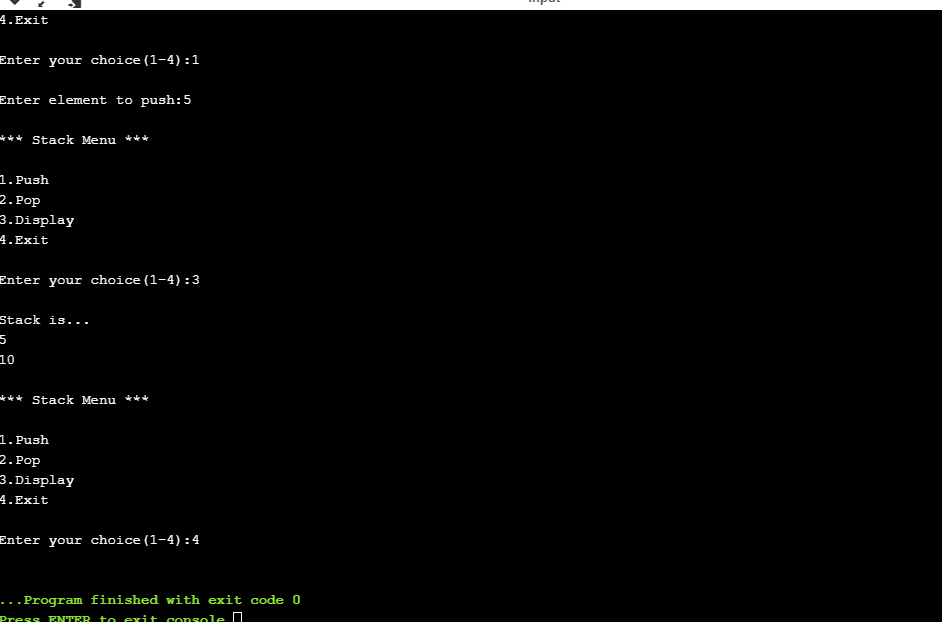
printf("\nStack is...\n");

for(i=top;i>=0;--i)

printf("%d\n",stack[i]);

}

}



**8. C program to perform expression conversion and evaluation. Take infix expression as run-time**

**input. Convert the expression into postfix and evaluate the postfix expression.**

**Let,x is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression y.**

1. **Push “(“onto Stack, and add “)” to the end of X.**
2. **Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.**
3. **If an operand is encountered, add it to Y.**
4. **If a left parenthesis is encountered, push it onto Stack.**
5. **If an operator is encountered ,then:**
   1. **Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.**
   2. **Add operator to Stack.  
      [End of If]**
6. **If a right parenthesis is encountered ,then:**
   1. **Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.**
   2. **Remove the left Parenthesis.  
      [End of If]  
      [End of If]**
7. **END.**

#define SIZE 50

#include <ctype.h>

#include <stdio.h>

char s[SIZE];

int top = -1;

void RemoveSpaces(char\* source) {

char\* i = source;

char\* j = source;

while(\*j != 0) {

\*i = \*j++;

if(\*i != ' ')

i++;

}

\*i = 0;

}

void push(char elem) {

s[++top] = elem;

}

char pop() {

return (s[top--]);

}

int pr(char elem) {

switch (elem) {

case '#':

return 0;

case '(':

return 1;

case '+':

case '-':

return 2;

case '\*':

case '/':

return 3;

}

}

void infix\_to\_postfix(char \*infix, char \*postfix) {

char ch, elem;

int i = 0, k = 0;

RemoveSpaces(infix);

push('#');

while ((ch = infix[i++]) != '\n') {

if (ch == '(')

push(ch);

else if (isalnum(ch))

postfix[k++] = ch;

else if (ch == ')') {

while (s[top] != '(')

postfix[k++] = pop();

elem = pop();

} else {

while (pr(s[top]) >= pr(ch))

postfix[k++] = pop();

push(ch);

}

}

while (s[top] != '#')

postfix[k++] = pop();

postfix[k] = 0;

}

int eval\_postfix(char \*postfix) {

char ch;

int i = 0, op1, op2;

while((ch = postfix[i++]) != 0) {

if(isdigit(ch))

push(ch-'0');

else {

op2 = pop();

op1 = pop();

switch(ch) {

case '+' : push(op1+op2);

break;

case '-' : push(op1-op2);

break;

case '\*' : push(op1\*op2);

break;

case '/' : push(op1/op2);

break;

}

}

}

return s[top];

}

void main() {

char infx[50], pofx[50];

printf("\nInput the infix expression: ");

fgets(infx, 50, stdin);

infix\_to\_postfix(infx, pofx);

printf("\nGiven Infix Expression: %sPostfix Expression: %s", infx, pofx);

top = -1;

printf("\nResult of evaluation of postfix expression : %d", eval\_postfix(pofx));

}

