**CODE:-**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct node node;

struct node

{

char info[100];

node \*link;

} \*top = NULL;

int isEmpty()

{

if (top == NULL)

return 1;

else

return 0;

}

void push(char \*item)

{

node \*temp;

temp = (node \*)malloc(sizeof(node));

if (temp == NULL)

{

printf("Memory cannot be allocated.\n");

exit(1);

}

strcpy(temp->info, item);

temp->link = top;

top = temp;

}

int stack\_count()

{

node \*p;

int count;

p = top;

while (p != NULL)

{

count++;

p = p->link;

}

return count;

}

char \*pop()

{

node \*temp;

char \*s = malloc(100);

if (isEmpty())

{

printf("Stack Underflow.\n");

return;

}

temp = top;

top = top->link;

strcpy(s, temp->info);

free(temp);

return s;

}

void display()

{

node \*p;

p = top;

printf("The elements of the stack are : ");

while (p != NULL)

{

printf("%s ", p->info);

p = p->link;

}

printf("\n");

}

void conv\_postfix\_to\_infix()

{

char postfix[200], infix[400], symbol[2], symb, a[200], b[200], tempstr[400] = "";

printf("Enter the postfix array.\n");

scanf("%s", postfix);

for (int i = 0; postfix[i] != '\0'; i++)

{

char symbol[2] = {postfix[i], '\0'};

symb = postfix[i];

switch (symb)

{

case '+':

case '-':

case '\*':

case '/':

case '%':

case '^':

if (stack\_count() < 2)

{

printf("Error.Not sufficient values in the expression.\n");

exit(1);

}

strcpy(a, pop());

strcpy(b, pop());

sprintf(tempstr, "%s %s %s %s %s", "(", b, symbol, a, ")");

push(tempstr);

break;

default:

push(symbol);

}

}

if (stack\_count() == 1)

printf("The required infix string is : %s\n", pop());

else

printf("Error.The user input has too many values.\n");

top = NULL;

}

int main()

{

int ch;

char cr;

while (1)

{

printf("Enter 1 to push an element.\n");

printf("Enter 2 to pop an element.\n");

printf("Enter 3 to display elements in stack.\n");

printf("Enter 4 to convert postfix to infix.\n");

printf("Enter 5 to exit.\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter the character to be inserted.\n");

getchar();

scanf("%c", &cr);

push(&cr);

break;

case 2:

printf("Enter 2 to delete an element.\n");

printf("The deleted element was %s.\n", pop());

break;

case 3:

display();

break;

case 4:

top = NULL;

conv\_postfix\_to\_infix();

break;

case 5:

exit(1);

default:

printf("Erroneous input.\n");

}

}

}

**OUTPUT:-**

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

1

Enter the character to be inserted.

3

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

1

Enter the character to be inserted.

5

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

1

Enter the character to be inserted.

a

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

3

The elements of the stack are : a 5 3

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

2

Enter 2 to delete an element.

The deleted element was a.

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

3

The elements of the stack are : 5 3

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

4

Enter the postfix array.

efg-+he-sh-o+/\*

The required infix string is : ( ( e + ( f - g ) ) \* ( ( h - e ) / ( ( s - h ) + o ) ) )

Enter 1 to push an element.

Enter 2 to pop an element.

Enter 3 to display elements in stack.

Enter 4 to convert postfix to infix.

Enter 5 to exit.

5