***ASSIGNMENT -2***

1. **WAP to implement Stack.  
    a. Push an element onto a stack.  
    b. Pop the stack.  
    c. Display the contents of the stack.**

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

struct stack{

int arr[SIZE];

int top;

};

struct stack s;

void push(int item);

int pop();

void display();

int main()

{

s.top = -1;

int c,data;

while(1)

{

printf("Enter what you want to do\n");

printf("1.Push an element in stack\n");

printf("2.Pop the stack\n");

printf("3.Display the stack\n");

printf("4.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:

{

printf("\n\nEnter the element to be pushed\n");

scanf("%d",&data);

push(data);

printf("\n\n");

break;

}

case 2:

{

printf("\nElement which is popped:%d\n",pop());

break;

}

case 3:

{

display();

break;

}

case 4:

exit(1);

}

}

return 0;

}

void push(int item)

{

if((s.top) == SIZE-1)

printf("Stack overflow\n");

else

s.arr[++(s.top)] = item;

}

int pop()

{

if((s.top) == -1)

printf("Stack underflow\n");

else

return s.arr[(s.top)--];

}

void display()

{

int i;

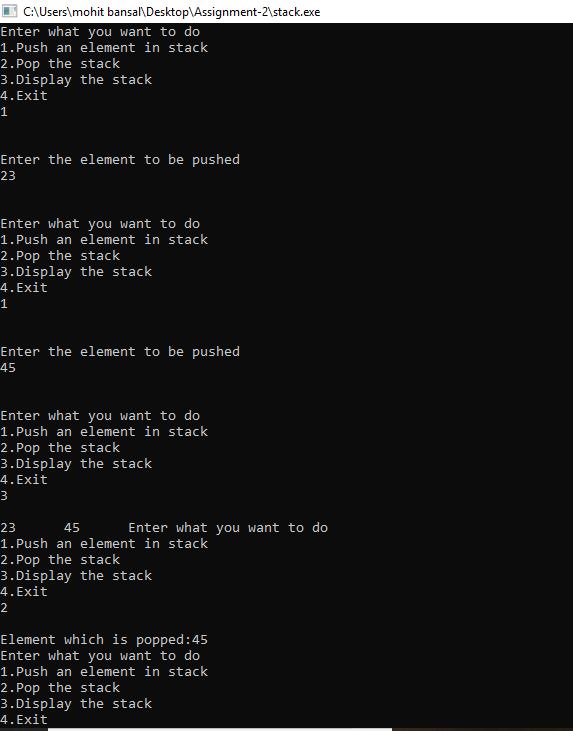
printf("\n");

for(i=0;i<=s.top;i++)

printf("%d\t",s.arr[i]);

}

**OUTPUT :**



1. **WAP to convert the following expressions:**

**a. Infix to Postfix**

#include<stdio.h>

#include<string.h>

#define SIZE 20

struct stack{

char str[SIZE];

int top;

}s;

void push(char ch);

char pop();

int pre(char ch);

char asso(char ch);

int main()

{

char ch;

s.top = -1;

while(1)

{

scanf("%c",&ch);

if(ch == '\n')

break;

if((ch>='a' && ch<='z') || (ch>='A' && ch <='Z'))

printf("%c",ch);

else

{

if(s.top == -1)

push(ch);

else if(ch == '(')

push(ch);

else if(ch == ')')

{

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

printf("%c",pop());

s.top--;

}

else

{

if(pre(ch) > pre(s.str[s.top]))

push(ch);

else if(pre(ch) == pre(s.str[s.top]))

{

if(asso(ch) == 'l')

{

printf("%c",pop());

push(ch);

}

if(asso(ch) == 'r')

push(ch);

}

else

{

while(pre(s.str[s.top]) >= pre(ch))

printf("%c",pop());

push(ch);

}

}

}

}

while(s.top >=0)

printf("%c",pop());

return 0;

}

void push(char ch)

{

s.str[++(s.top)] = ch;

}

char pop()

{

return s.str[(s.top)--];

}

int pre(char ch)

{

switch(ch)

{

case '(' :

return 1;

case '+' :

case '-' :

return 2;

case '/' :

case '\*' :

return 3;

case '^' :

return 4;

}

}

char asso(char ch)

{

switch(ch)

{

case '+' :

case '-' :

case '/' :

case '\*' :

return 'l';

case '^' :

return 'r';

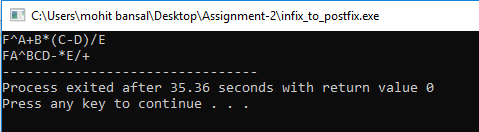
case '(' :

return 'r';

}

}

**OUTPUT :**



**b. Infix to Prefix**

#include<stdio.h>

#include<string.h>

#define SIZE 50

struct stack{

char str[SIZE];

int top;

}s;

char res[SIZE],temp[SIZE];

int cnt = 0;

void push(char ch);

char pop();

int pre(char ch);

char asso(char ch);

//void \*strrev(char str[]); //for linux

int main()

{

char ch[SIZE];

s.top = -1;

scanf("%s",ch);

strrev(ch);

int i,l=strlen(ch);

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

{

if((ch[i]>='a' && ch[i]<='z') || (ch[i]>='A' && ch[i]<='Z'))

res[cnt++] = ch[i];

else

{

if(s.top == -1)

push(ch[i]);

else if(ch[i] == ')')

push(ch[i]);

else if(ch[i] == '(')

{

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

res[cnt++] = pop();

s.top--;

}

else

{

if(pre(ch[i]) > pre(s.str[s.top]))

push(ch[i]);

else if(pre(ch[i]) == pre(s.str[s.top]))

{

if(asso(ch[i]) == 'l')

{

res[cnt++] = pop();

push(ch[i]);

}

if(asso(ch[i]) == 'r')

push(ch[i]);

}

else

{

while(pre(s.str[s.top]) >= pre(ch[i]))

res[cnt++] = pop();

push(ch[i]);

}

}

}

}

while(s.top >=0)

res[cnt++] = pop();

strrev(res);

printf("%s",res);

return 0;

}

void push(char ch)

{

s.str[++(s.top)] = ch;

}

char pop()

{

return s.str[(s.top)--];

}

int pre(char ch)

{

switch(ch)

{

case ')' :

return 1;

case '+' :

case '-' :

return 2;

case '/' :

case '\*' :

return 3;

case '^' :

return 4;

}

}

char asso(char ch)

{

switch(ch)

{

case '+' :

case '-' :

case '/' :

case '\*' :

return 'l';

case '^' :

return 'r';

case ')' :

return 'r';

}

}

/\*

void \*strrev(char str[])

{

int i=strlen(str);

int j,k;

char c;

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

{

c = str[j];

str[j]=str[k];

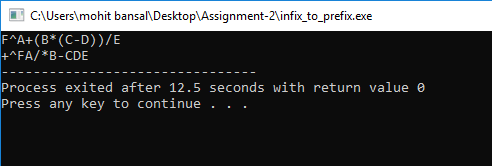
str[k] = c;

}

}

\*/

**OUTPUT :**



**c. Postfix to Infix**

#include<stdio.h>

#include<string.h>

#define SIZE 10

struct stack{

char str[SIZE][SIZE];

int top;

}s;

char con[2] = "\0";

void push(char a[]);

char \*pop();

char \*convert(char c);

int main()

{

char res[SIZE]="\0",ch,temp[SIZE];

while(1)

{

scanf("%c",&ch);

if(ch=='\n')

break;

if((ch>='A' && ch<='Z')||(ch>='a' && ch<='z'))

push(convert(ch));

else

{

strcat(res,"(");

strcpy(temp,pop());

strcat(res,pop());

strcat(res,convert(ch));

strcat(res,temp);

strcat(res,")");

push(res);

}

strcpy(res,"\0");

}

printf("%s",pop());

return 0;

}

void push(char a[])

{

strcpy(s.str[++(s.top)],a);

}

char \*pop()

{

return s.str[(s.top)--];

}

char \*convert(char ch)

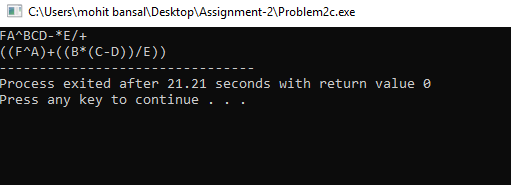
{

con[0] = ch;

return con;

}

**OUTPUT :**



**d. Postfix to Prefix.**

#include<stdio.h>

#include<string.h>

#define SIZE 50

struct stack{

char str[SIZE][SIZE];

int top;

}s;

char con[2]="\0";

void push(char a[]);

char \*pop();

char \*convert(char c);

int main()

{

char ch,res[SIZE]="\0",temp[SIZE];

while(1)

{

scanf("%c",&ch);

if(ch == '\n')

break;

if((ch >='a' && ch<='z') || (ch>='A' && ch<='Z'))

push(convert(ch));

else

{

strcpy(res,convert(ch));

strcpy(temp,pop());

strcat(res,pop());

strcat(res,temp);

push(res);

}

strcpy(res,"\0");

}

printf("%s",pop());

return 0;

}

void push(char a[])

{

strcpy(s.str[++(s.top)],a);

}

char \*pop()

{

return s.str[(s.top)--];

}

char \*convert(char ch)

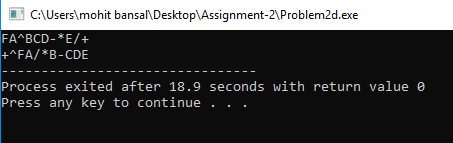
{

con[0] = ch;

return con;

}

**OUTPUT :**



**e. Prefix to Infix**

#include<stdio.h>

#include<string.h>

#define SIZE 50

struct stack{

char str[SIZE][SIZE];

int top;

}s;

char con[2] = "\0";

char str2[SIZE];

void push(char a[]);

char \*pop();

char \*convert(char c);

char \*convert2(char str[]);

int main()

{

char res[SIZE]="\0",ch,temp[SIZE];

char st[SIZE];

scanf("%s",st);

strrev(st);

int i,l=strlen(st);

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

{

if((st[i]>='A' && st[i]<='Z')||(st[i]>='a' && st[i]<='z'))

push(convert(st[i]));

else

{

strcat(res,"(");

strcpy(temp,pop());

strcat(res,pop());

strcat(res,convert(st[i]));

strcat(res,temp);

strcat(res,")");

push(res);

}

strcpy(res,"\0");

}

strcpy(st,pop());

strrev(st);

printf("%s",convert2(st));

return 0;

}

void push(char a[])

{

strcpy(s.str[++(s.top)],a);

}

char \*pop()

{

return s.str[(s.top)--];

}

char \*convert(char ch)

{

con[0] = ch;

return con;

}

char \*convert2(char a[])

{

int i;

strcpy(str2,a);

for(i=0;str2[i]!='\0';i++)

{

if(str2[i]=='(')

str2[i] = ')';

else if(str2[i]==')')

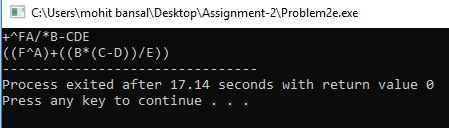
str2[i] = '(';

}

return str2;

}

**OUTPUT :**



**f. Prefix to Postfix.**

#include<stdio.h>

#include<string.h>

#define SIZE 50

struct stack{

char str[SIZE][SIZE];

int top;

}s;

char con[2]="\0";

void push(char a[]);

char \*pop();

char \*convert(char c);

int main()

{

char st[SIZE],res[SIZE]="\0",temp[SIZE];

scanf("%s",st);

strrev(st);

int i,l=strlen(st);

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

{

if(st[i] == '\n')

break;

if((st[i] >='a' && st[i]<='z') || (st[i]>='A' && st[i]<='Z'))

push(convert(st[i]));

else

{

strcpy(res,convert(st[i]));

strcpy(temp,pop());

strcat(res,pop());

strcat(res,temp);

push(res);

}

strcpy(res,"\0");

}

strcpy(st,pop());

strrev(st);

printf("%s",st);

return 0;

}

void push(char a[])

{

strcpy(s.str[++(s.top)],a);

}

char \*pop()

{

return s.str[(s.top)--];

}

char \*convert(char ch)

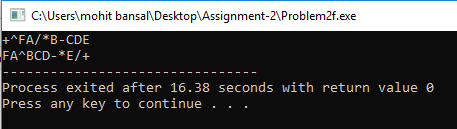
{

con[0] = ch;

return con;

}

**OUTPUT :**



1. **WAP to evaluate a postfix expression.**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#define SIZE 20

struct stack{

int arr[SIZE];

int top;

}s;

void push(int item);

int pop();

int main()

{

s.top = -1;

char ch;

while(1)

{

scanf("%c",&ch);

if(ch=='\n')

break;

if(ch == ' ')

continue;

if(isdigit(ch))

{

int num = 0;

while(isdigit(ch))

{

num = num\*10 + (int)(ch - '0');

scanf("%c",&ch);

}

push(num);

}

else

{

int a = pop();

int b = pop();

switch(ch)

{

case '+':

push(b+a);

break;

case '-':

push(b-a);

break;

case '\*':

push(b\*a);

break;

case '/':

push(b/a);

break;

case '^':

push(pow(b,a));

break;

}

}

}

printf("%d",pop());

return 0;

}

void push(int item)

{

s.arr[++(s.top)] = item;

}

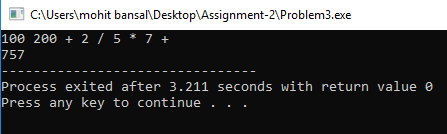
int pop()

{

return s.arr[(s.top)--];

}

**OUTPUT :**



1. **WAP to implement two stacks in a single array.**

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

struct stack{

int arr[SIZE];

int top1,top2;

};

struct stack s;

void push1(int item);

void push2(int item);

int pop1();

int pop2();

void display1();

void display2();

int main()

{

s.top1 = -1;

s.top2 = SIZE;

int data,c;

while(1)

{

printf("Enter your choice what you want to do\n");

printf("1.Push an element in stack1\n");

printf("2.Push an element in stack2\n");

printf("3.Pop the stack1\n");

printf("4.Pop the stack2\n");

printf("5.Display stack1\n");

printf("6.Display stack2\n");

printf("7.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:

{

printf("\nnter the element :");

scanf("%d",&data);

push1(data);

break;

}

case 2:

{

printf("\nenter the element :");

scanf("%d",&data);

push2(data);

break;

}

case 3:

{

printf("\nElement which is popped:%d",pop1());

break;

}

case 4:

{

printf("\nElement which is popped:%d",pop2());

break;

}

case 5:

{

display1();

printf("\n");

break;

}

case 6:

{

display2();

printf("\n");

break;

}

case 7:

exit(1);

}

}

return 0;

}

void push1(int item)

{

if(s.top2-s.top1==1)

printf("stack overflow\n");

else

s.arr[++(s.top1)] = item;

}

void push2(int item)

{

if(s.top2-s.top1==1)

printf("stack overflow\n");

else

s.arr[--(s.top2)] = item;

}

int pop1()

{

if(s.top1 == -1)

{

printf("stack1 underflow\n");

return -1;

}

return s.arr[(s.top1)--];

}

int pop2()

{

if(s.top2 == SIZE)

{

printf("stack2 underflow\n");

return -1;

}

return s.arr[(s.top2)++];

}

void display1()

{

int i;

for(i=0;i<=s.top1;i++)

printf("%d\t",s.arr[i]);

}

void display2()

{

int i;

for(i=SIZE-1;i>=s.top2;i--)

printf("%d\t",s.arr[i]);

}

**OUTPUT :**

