**EXPERIMENT 1 : Stack using array**

#include<stdio.h>

#include<stdlib.h>

struct stack

{

int size,top,\*s;

};

int create(struct stack \*st)

{

printf("entre size");

scanf ("%d",&st->size);

st->top=-1;

st->s=(int \*)malloc(st->size\*(sizeof(int)));

}

int push(struct stack\*st,int x)

{

if (st->top==st->size-1)

printf("stack over flow");

else

st->top++;

st->s[st->top]=x;

}

int pop(struct stack \*st)

{

int x=-1;

if(st->top==-1)

{

printf ("stack under flow");

}

else

x=st->s[st->top--];

return x;

}

void display(struct stack st)

{

int i;

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

printf ("%d\n",st.s[i]);

}

int main()

{

struct stack st;

create(&st);

push(&st,10);

push(&st,20);

printf ("delete element is %d\n",pop(&st));

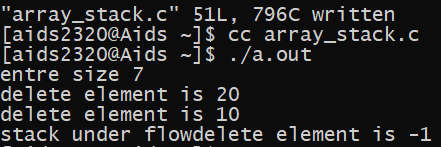
printf ("delete element is %d\n",pop(&st));

printf ("delete element is %d\n",pop(&st));

display(st);

}

OUTPUT:



EXPERIMENT 3(i):

#include<stdio.h>

#include<stdlib.h>

struct queue

{

int size;

int front;

int rear;

int \*Q;

};

create (struct queue \*q,int size)

{

q->size=size;

q->front=q->rear=-1;

q->Q=(int\*)malloc(q->size\*sizeof(int));

}

void enque(struct queue\*q,int X)

{

if (q->rear==q->size-1)

printf("queue is full");

else

{

q->rear++;

q->Q[q->rear]=X;

}

}

int dequee (struct queue \*q)

{

int x=-1;

if (q->front==q->rear)

printf("queue is empty") ;

else

{

q->front++;

x=q->Q[q->front];

}

return x;

}

void display (struct queue q)

{

int i;

for(i=q.front +1;i<=q.rear;i++)

printf ("%d",q.Q[i]);

}

int main()

{

struct queue q;

create(&q,5);

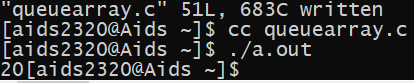
enque(&q,20);

display(q);

dequee(&q);

}

OUTPUT:



EXPERIMENT 4:

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

//function to return precedence of operators

int prec(char c)

{

if(c=='^')

return 3;

else if (c=='/'||c=='\*')

return 2;

else if (c=='+'||c=='-')

return 1;

else

return-1;

}

//function to perform infix to postfix conversion

void infixtopostfix(char\*exp)

{

int len=strlen(exp);

char result[len+1];

char stack[len];

int j=0;

int top=-1;

int i;

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

{

char c=exp[i];

//if the scanned charecter is

//an operand,add it to the output string

if (isalnum(c))

result [j++]=c;

//if the scanned character is

//an '(',push it to the stack

else if (c=='(')

stack[++top]='(';

//if the scanned charecter is an')',

//pop and add to the output string from the stack

//until an'('is encountered

else if(c==')')

{

while (top!=-1&&stack[top]!='(')

{

result [j++]=stack[top--];

}

top--;

}

//if an operator is scanned

else

{

while (top!=-1&&(prec(c)<prec(stack[top])||prec(c)==prec(stack[top])))

{

result [j++]=stack [top--];

}

stack[++top]=c;

}

}

//pop all the remaining elements from the stack

while (top!=-1)

{

result [j++]=stack[top--];

}

result[j]='\0';

printf("%s\n",result);

}

int main()

{

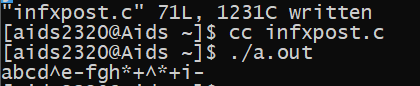
char exp[]="a+b\*(c^d-e)^(f+g\*h)-i";

infixtopostfix(exp);

return 0;

}

OUTPUT:



EXPERIMENT 5 :

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

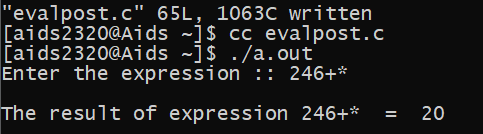
}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

return 0;

}

OUTPUT:



EXPERIMENT 6 :

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}

\*top=NULL;

void push(int x)

{

struct node \*t;

t=(struct node \*)malloc(sizeof(struct node));

if(t==NULL)

printf("stack is full");

else

t->data=x;

t->next=top;

top=t;

}

int pop()

{

struct node \*t;

int x=-1;

if(top==NULL)

printf("stack is empty");

else

t=top;

top=top->next;

x=t->data;

free(t);

return x;

}

void display()

{- INSERT -- 2,2 4%

struct node \*p;

p=top;

while(p!=NULL)

{

printf("%d\n",p->data);

p=p->next;

}

}

int main()

{

push(99);

push(88);

push(100);

push(110);

display();

printf("deleted element is%d\n",pop());

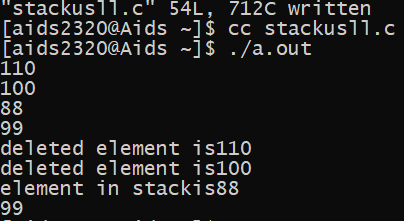
printf("deleted element is%d\n",pop());

printf("element in stackis");

Display();

}

OUTPUT :



EXPERIMENT 7 :

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}

\*front=NULL,\*rear=NULL;

void enqueue (int x)

{

struct node \*t;

t=(struct node\*)malloc(sizeof(struct node));

if(t==NULL)

printf("quequ is full");

else

{

t->data=x;

t->next=NULL;

if(front==NULL)

front=rear=t;

else

{

rear->next=t;

rear=t;

}

}

}

int dequeue()

{

int x=-1;

struct node\*t;

if(front==NULL)

{

printf ("queue is full");

}

else

{

x=front->data;

t=front;

front=front->next;

free(t);

}

return x;

}

void Display()

{

struct node\*p=front;

while(p!=NULL)

{

printf("%d",p->data);

p=p->next;

}

}

int main()

{

enqueue(10);

enqueue(20);

printf("printed element is %d",dequeue());

return 0;

}

OUTPUT:

