1. Write a C program to implement single linked list with the following operations

case1: insert an element in the list

case 2: delete an element from the list and display the values

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insert(struct Node\*\* head, int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void deleteElement(struct Node\*\* head, int value) {

if (\*head == NULL) {

printf("List is empty.\n");

return;

}

struct Node\* current = \*head;

struct Node\* previous = NULL;

while (current != NULL && current->data != value) {

previous = current;

current = current->next;

}

if (current == NULL) {

printf("Element not found in the list.\n");

return;

}

if (previous == NULL) {

\*head = current->next;

} else {

previous->next = current->next;

}

free(current);

printf("Element deleted successfully.\n");

}

void display(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

printf("\n");

}

int main() {

struct Node\* head = NULL;

int choice, value;

while (1) {

printf("1. Insert an element\n");

printf("2. Delete an element\n");

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

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to insert: ");

scanf("%d", &value);

insert(&head, value);

break;

case 2:

printf("Enter the value to delete: ");

scanf("%d", &value);

deleteElement(&head, value);

break;

case 3:

display(head);

break;

case 4:

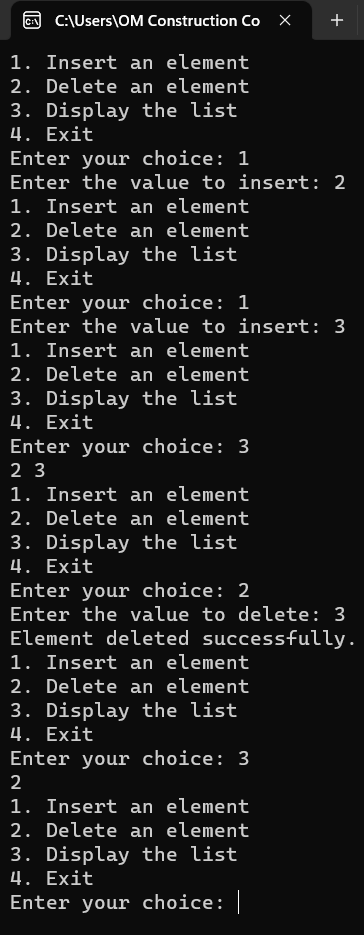
exit(0);

default:

printf("Invalid choice. Please try again.\n");}}

return 0;

}



1. Write a C program to implement stack data structure push an element ,pop an element and display an element from the stack.

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 5

struct stack

{

int stk[MAXSIZE];

int top;

};

typedef struct stack ST;

ST s;

void push ()

{

int num;

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

{

printf ("Stack is Full\n");

return;

}

else

{

printf ("\nEnter element to be pushed : ");

scanf ("%d", &num);

s.top = s.top + 1;

s.stk[s.top] = num;

}

return;

}

int pop ()

{

int num;

if (s.top == - 1)

{

printf ("Stack is Empty\n");

return (s.top);

}

else

{

num = s.stk[s.top];

printf ("poped element is = %d\n", s.stk[s.top]);

s.top = s.top - 1;

}

return(num);

}

void display ()

{

int i;

if (s.top == -1)

{

printf ("Stack is empty\n");

return;

}

else

{

printf ("\nStatus of elements in stack : \n");

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

{

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

}

}

}

int main ()

{

int ch;

s.top = -1;

printf ("\tSTACK OPERATIONS\n");

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

printf(" 1. PUSH\n");

printf(" 2. POP\n");

printf(" 3. DISPLAY\n");

printf(" 4. EXIT\n");

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

while(1)

{

printf("\nChoose operation : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

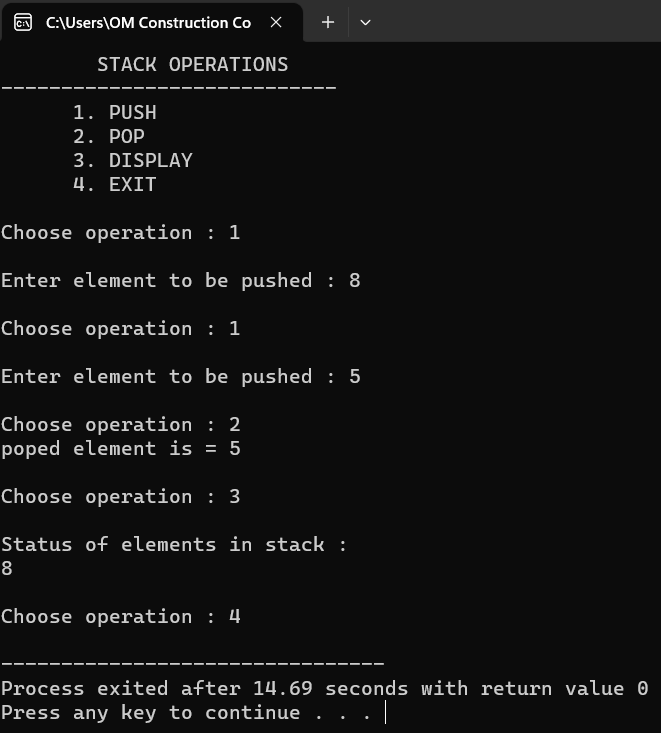
printf("Invalid operation \n");

}

}

return 0;

}



1. Write a C program to implement queue as in follow eneque ,dequeue ,and display

#include<stdio.h>

#define SIZE 3

struct queue

{

int values[SIZE];

int front;

int rear;

};

void enqueue(int);

int dequeue();

int isEmpty();

int isFull();

void display();

struct queue q;

int main()

{

q.front = -1;

q.rear = -1;

int user\_choice, data;

char user\_active = 'Y';

while (user\_active == 'Y' || user\_active == 'y')

{ printf("\n--------Queue Program------\n");

printf("\n1. Enqueue");

printf("\n2. Dequeue");

printf("\n3. Display");

printf("\n4. Exit");

printf("\n\nEnter your choice: ");

scanf("%d", &user\_choice);

switch(user\_choice)

{ case 1:

printf("\nEnter Data: ");

scanf("%d", &data);

enqueue(data);

break;

case 2:

if (!isEmpty())

{

data = dequeue();

printf("\n\* %d was removed!\n", data);}

else

{ printf("\nQueue is Empty!\n");}

break;

case 3:

display();

printf("\n");

break;

case 4:

printf("\n\tProgram was terminated!\n\n");

return 1;

default:

printf("\n\tInvalid Choice\n");

} printf("\nDo You want to continue? ");

fflush(stdin);

scanf(" %c", &user\_active); }

return 0;

} int isEmpty()

{

if (q.front == -1 || q.front > q.rear)

{ return 1; }

return 0; }

int isFull()

{

if (q.rear == SIZE - 1)

{ return 1; }

return 0; }

void enqueue(int data)

{ if (isFull()) {

printf("\nQueue is Full!\n");

return;

} if (isEmpty())

{ q.front += 1; }

q.rear += 1;

q.values[q.rear] = data;

printf("\n\* %d was inserted!\n", data);

} int dequeue()

{ if (!isEmpty())

{ int data = q.values[q.front];

q.front += 1;

return data;}}

void display()

{ if (isEmpty()){

printf("\nQueue is Empty\n");

return;

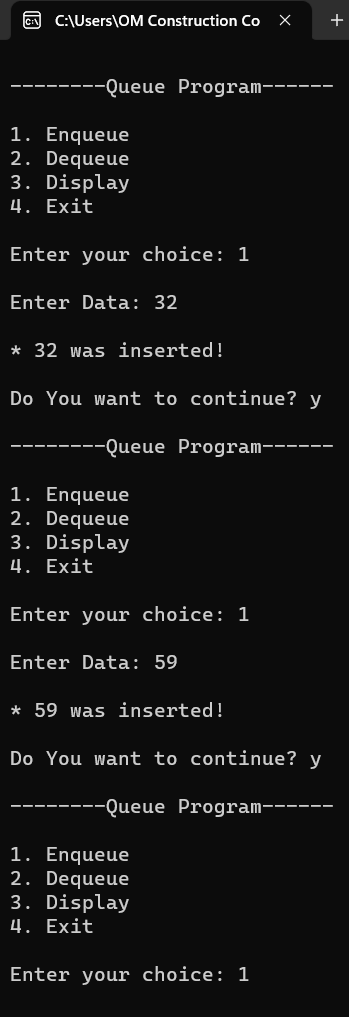
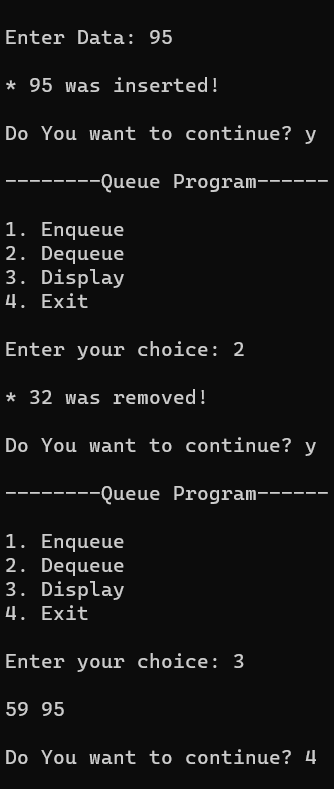
} printf("\n");

int begin = q.front;

while (begin <= q.rear)

{ printf("%d ", q.values[begin]);

begin += 1; }}

1. Write a C program to convert infix expression into postfix using stack.

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{ stack[++top] = x; }

char pop()

{ if(top == -1)

return -1;

else

return stack[top--]; }

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

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

{ if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{ while((x = pop()) != '(')

printf("%c ", x);}

else

{ while(priority(stack[top]) >= priority(\*e))

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

push(\*e);

}

e++;

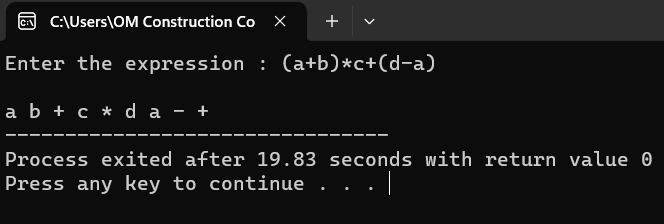
} while(top != -1)

{

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

}return 0;

}



1. Write a C program to evaluate the given expression using stack.

#include <stdio.h>

#include <ctype.h>

#define MAXSTACK 100

#define POSTFIXSIZE 100

int stack[MAXSTACK];

int top = -1;

void push(int item)

{ if (top >= MAXSTACK - 1) {

printf("stack over flow");

return;

}

else {

top = top + 1;

stack[top] = item;

} }

int pop()

{

int item;

if (top < 0) {

printf("stack under flow");

}

else {

item = stack[top];

top = top - 1;

return item;

} } void EvalPostfix(char postfix[])

{ int i;

char ch;

int val;

int A, B;

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

ch = postfix[i];

if (isdigit(ch)) {

push(ch - '0');

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {

A = pop();

B = pop();

switch (ch)

{

case '\*':

val = B \* A;

break;

case '/':

val = B / A;

break;

case '+':

val = B + A;

break;

case '-':

val = B - A;

break; }

push(val); } }

printf(" \n Result of expression evaluation : %d \n", pop());

}

int main()

{ int i;

char postfix[POSTFIXSIZE];

printf(" There are only four operators(\*, /, +, -) in an expression and operand is single digit only.\n");

printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

for (i = 0; i <= POSTFIXSIZE - 1; i++) {

scanf("%c", &postfix[i]);

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

{

break;

}

} EvalPostfix(postfix);

return 0;

}

