**1.Write a program to implement stack using array.**

#include<stdio.h>

#include<stdlib.h>

int n, top = -1, \*stack;

void push(int x){

if(top==n) return;

stack[++top]=x;

}

int pop(){

if(top==-1) return -1;

return stack[top--];

}

int peek(){

if(top==-1) return -1;

return stack[top];

}

void display(){

for(int i=top ; i>-1 ; i--) printf("%d ",stack[i]);

printf("\n\n");

}

int main(){

n = 10;

printf("Initializing the stack with size 10\n\n");

stack = (int\*)malloc(n\*sizeof(int));

printf("Pushing elements into the stack\n1\n2\n3\n\n");

push(1);

push(2);

push(3);

printf("Displaying elements of the stack -\n");

display();

printf("The top of the stack = %d\n\n",peek());

printf("Pop the top of the stack = %d\n\n",pop());

printf("Pop the top of the stack = %d\n\n",pop());

printf("Displaying elements of the stack -\n");

display();

return 0;

}

**Output:**

Initializing the stack with size 10

Pushing elements into the stack

1

2

3

Displaying elements of the stack -

3 2 1

The top of the stack = 3

Pop the top of the stack = 3

Pop the top of the stack = 2

Displaying elements of the stack -

1

**2.Write a program to implement stack using linked list**.

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

Node\* top = NULL;

void push(int value) {

struct Node \*newNode;

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

newNode->data = value; if (top == NULL) {

newNode->next = NULL;

} else {

newNode->next = top;

}

top = newNode;

printf("Node is Inserted\n\n");

}

int pop() {

if (top == NULL) {

printf("\nStack Underflow\n");

} else {

struct Node \*temp = top;

int temp\_data = top->data;

top = top->next;

free(temp);

return temp\_data;

}

}

void display() {

if (top == NULL) {

printf("\nStack Underflow\n");

} else {

printf("The stack is \n");

struct Node \*temp = top;

while (temp->next != NULL) {

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

temp = temp->next;

}

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

}

}

int main() {

int choice, value;

printf("\nImplementation of Stack using Linked List\n");

while (1) {

printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("\nEnter the value to insert: ");

scanf("%d", &value);

push(value);

break;

case 2:

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

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nWrong Choice\n");

}

}

}

**3.Write a program to check whether number of opening ‘(‘ & closing ‘)’ parenthesis are**

**equal or not.**

#include <stdio.h>

#include <stdlib.h>

#define bool int

struct sNode {

char data;

struct sNode\* next;

};

void push(struct sNode\*\* top\_ref, int new\_data);

int pop(struct sNode\*\* top\_ref);

bool isMatchingPair(char character1, char character2){

if (character1 == '(' && character2 == ')')

return 1;

else if (character1 == '{' && character2 == '}')

return 1;

else if (character1 == '[' && character2 == ']')

return 1;

else

return 0;

}

bool areBracketsBalanced(char exp[])

{

int i = 0;

struct sNode\* stack = NULL;

while (exp[i]) {

if (exp[i] == '{' || exp[i] == '(' || exp[i] == '[')

push(&stack, exp[i]);

if (exp[i] == '}' || exp[i] == ')'

|| exp[i] == ']') {

if (stack == NULL)

return 0;

else if (!isMatchingPair(pop(&stack), exp[i]))

return 0;

}

i++;

}

if (stack == NULL)

return 1;

else

return 0; }

int main()

{

char exp[100] = "{()}[]";

if (areBracketsBalanced(exp))

printf("Balanced \n");

else

printf("Not Balanced \n");

return 0;

}

void push(struct sNode\*\* top\_ref, int new\_data){

struct sNode\* new\_node

= (struct sNode\*)malloc(sizeof(struct sNode));

if (new\_node == NULL) {

printf("Stack overflow n");

getchar();

exit(0);

}

new\_node->data = new\_data;

new\_node->next = (\*top\_ref);

(\*top\_ref) = new\_node;

}

int pop(struct sNode\*\* top\_ref) {

char res;

struct sNode\* top;

if (\*top\_ref == NULL) {

printf("Stack overflow n");

getchar();

exit(0);

}

else {

top = \*top\_ref;

res = top->data;

\*top\_ref = top->next;

free(top);

return res;

}

}

**Output:** Balanced

**4.Write a program to convert infix expression to postfix expression.**

#include <stdio.h>

#include<ctype.h>

#define SIZE 100

char stack[SIZE];

int tos=-1;

void push(char x) {

stack[++tos]=x;

}

char pop() {

if(tos==-1) {

printf("\nUnderflow");

return -1;

}

else {

return stack[tos--];

}

}

int priority(char x) {

if(x=='('){

return 0;

}

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

return 1;

}

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

return 2;

}

}

int main() {

char ex[100];

char \*e,x;

printf("Put the expression:\n");

scanf("%s",ex);e=ex;

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[tos])>=priority(\*e)) {

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

}

push(\*e);

}

e++;

}

while(tos!=-1) {

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

}

}

**Output:**

Put the expression:

(a+b-\*c/)

a b + c \* / -

**5.Write a program to evaluate a postfix expression**.

#include <stdio.h>

#define SIZE 100

int stack[SIZE];

int tos =-1;

void push(int x){

stack[++tos]=x;

}

void pop() {

if(stack[])

return stack[tos--];

}

int main()

{

char ex[100];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression:\n");

scanf("%d",&ex);

e=ex;

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

if(isdigit(\*e)) {

num=\*e-48;

}

else {

n1=pop();

n2=pop();

switch(\*e) {

case '+':

n3=n1+n2;

break;

case '-':

n3=n2-n1;

break;

case '\*':

n3=n1\*n2;

break;case '/':

n3=m2/n1;

break;

}

push(n3);

}

e++;

}

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

}

**6.Write a program to implement recursive & tail recursive version a) to find factorial of a number b) to find Fibonacci series.**

**//a)**

#include <stdio.h>int factorial();

int main() {

printf("Factorial is: %d",factorial());

}

int factorial() {

int i,n,fact=1;

printf("Enter the limit:\n");

scanf("%d",&n);

for(i=1;i<=n;i++){

fact=fact\*i;

}

return fact;

}

**Output:**

Enter the limit:

5

Factorial is: 120

**//b)**

#include <stdio.h>

int fibonacci(int);

int main() {

int a=0,b=1,n,c,i;

printf("Enter the limit: ");

scanf("%d",&n);

printf("Fibonacci series is:\n");

printf("%d %d",a,b);

for(i=1;i<=n;i++){

printf("%d ", fibonacci(n));

}

}

int fibonacci(int n) {

int c,a=0,b=1;

c=a+b;

printf(" %d ",c);

a=b;

b=c;

}

**Output:**

Enter the limit: 4

Fibonacci series is:

0 1 1 1 1 1 1 1 1 1

**7.Write a program to implement ‘Tower of Hanoi’ using recursion.**

#include<stdio.h>

void TOH(int n,char x,char y,char z) {

if(n>0) {

TOH(n-1,x,z,y);

printf("\n%c to %c",x,y);

TOH(n-1,z,y,x);

}

}

int main() {

int n=3;

TOH(n,'A','B','C');

}

**Output:**

A to B

A to C

B to C

A to B

C to A

C to B

A to B

**8.Write a program to convert a decimal number into binary number using stack.**

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

int isEmpty(int top, int stack\_arr[]);

void push(int x, int \*top, int stack\_arr[]);

int pop(int \*top, int stack\_arr[]);

void DecToBin(int num);

int main(){

int num;

printf("Enter an integer : ");

scanf("%d",&num);

printf("Binary Equivalent is : ");

DecToBin(num);

return 0;

}

void DecToBin(int num){

int stack[MAX], top=-1, rem;

while(num!=0) {

rem = num%2;

push(rem, &top, stack);

num/=2;

}

while(top!=-1)

printf("%d", pop(&top, stack));

printf("\n");

}

void push(int x, int \*top, int stack\_arr[]){

if(\*top == (MAX-1))

printf("Stack Overflow\n");

else {

\*top=\*top+1;

stack\_arr[\*top] = x;

}

}

int pop(int \*top, int stack\_arr[]){

int x;

if(\*top == -1){

printf("Stack Underflow\n");

exit(1);

}

else {

x = stack\_arr[\*top];

\*top=\*top-1;

}

return x;

}

**Output:**

Enter an integer : 6

Binary Equivalent is : 110

**9.Write a program to reverse a string using stack.**

#include <stdio.h>

#include<string.h>

#define SIZE 50

int stack[SIZE];

int tos=-1;

void push(char x)

{

stack[++tos]=x;

}

void pop()

{

printf("%c",stack[tos--]);

}

int main()

{

char str[]=”hello";

int len=strlen(str);

int i;

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

{

push(str[i]);

}

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

{

pop();

}

}

**Output:** olleh

1. **Write a program to implement multiple stacks in a single array.**

#include <stdio.h>

#include <stdlib.h>

int s[50],top[50],min[50],max[50];

int ns,size;

void init(void) {

int i;

for(i=0;i<50;++i) {

s[i]=min[i]=max[i] = 0;

top[i]=-1;

}

}

void createstack() {

int i ;

min[0]= -1;

max[0] = size -1;

top[0]=-1;

for(i=1;i<ns;++i) {

min[i]= min[i-1] + size;

top[i] = min [i];

}

for(i=1;i<ns;++i) {

max[i]= min[i+1];

}

}

void push(int ele,int k)

{

if(top[k-1]==max[k-1])

{

printf(“Stack no %d is full i.e overflow\n”,k);

return;

}

++top[k-1];

s[top[k-1]] = ele;

}

void pop(int k){

if(top[k-1]==min[k-1]) {

printf(“\nStack no %d is empty i.e underflow\n”,k);

return;

}

printf(“%d from stack %d is deleted:\n”,s[top[k-1]],k);

- -top[k-1];

}

void display(int k){

int j;

if(top[k-1]==min[k-1]) {

printf(“\nStack no %d is empty\n”,k);

return;

}

printf(“\nStack %d →> “,k);

for(j=min[k-1]+1;j<=top[k-1];++j){

printf(“%d”,s[j] );

}

}

int main() {

int ele,ch,skn;

init();

printf(“\nEnter the number of Stacks\n”);

scanf(“%d”,&ns);

size = 50/ns;

createstack();

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

do{

printf(“\nEnter your choice : \t”);

scanf(“%d”,&ch);

switch(ch) {

case 1: printf(“\nEnter the stack no : \t”);

scanf(“%d”,&skn);

printf(“\nEnter the element : \t”);

scanf(“%d”,&ele);

push(ele,skn);

break;

case 2 : printf(“\nEnter the stack no to pop : \t”);

scanf(“%d”,&skn);

pop(skn);

break;

case 3: printf(“\nEnter the stack no to display : \t”);

scanf(“%d”,&skn);

display(skn);

break;

case 4 : printf(“\nProgram Terminating”);

break;

default : printf(“\nInvalid Option\n”);

}

}

while(ch!=4);

return 0;

}