EXP1

#include <stdio.h>

int main() {

int a[100], size, i, loc, n, element; printf("Enter size: ");

scanf("%d", &size);

printf("Enter %d elements: ", size);

for (i = 0; i < size; i++) scanf("%d", &a[i]);

printf("List before Insertion: ");

for (i = 0; i < size; i++) printf("%d ", a[i]);

printf("\n");

printf("Enter element to insert: ");

scanf("%d", &element);

printf("Enter position: ");

scanf("%d", &loc);

loc--;

for (i = size; i > loc; i--) a[i] = a[i - 1];

a[loc] = element;

size++;

printf("List after Insertion: ");

for (i = 0; i < size; i++) printf("%d ", a[i]);

printf("\n");

printf("Enter element to delete: ");

scanf("%d", &n);

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

if (a[i] == n) {

for (int j = i; j < size - 1; j++) a[j] = a[j + 1];

size--;

break; }}

printf("List after deletion: ");

for (i = 0; i < size; i++) printf("%d ", a[i]);

printf("\n");

return 0;}

EXP 2

#include <stdio.h>

int sum(int n);

int main() {

int number, result;

printf("Enter a positive integer: ");

scanf("%d", &number);

result = sum(number);

printf("sum = %d", result);

return 0;

}int sum(int n) {

if (n != 0)

// sum() function calls itself

return n + sum(n-1);

else

return n;

}

(\*)EXP3

#include<stdio.h>

int top=-1,a,st[100];

void push();

void pop();

void peek();

void display();

void main(){

    int choice=0;

    printf("Enter the size of stack:");

    scanf("%d",&a);

    while(choice!=5){

        printf("Operations on Stack\n1.Push\n2.Pop\n3.Peek\n4.Display\n5.Exit");

        printf("\nEnter your choice:");

        scanf("%d",&choice);

        switch(choice){

            case 1:

            push();

            break;

            case 2:

            pop();

            break;

            case 3:

            peek();

            break;

            case 4:

            display();

            break;

            case 5:

            printf("Exiting...\n");

            break;

            default:

            printf("Invalid choice.Please enter a valid chocie.\n");

        }

    }

}

void push(){

    int val;

    if(top==(a-1)){

        printf("Stack Overflow\n");

    }

    else{

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

        scanf("%d",&val);

        top++;

        st[top]=val;

    }

}

void pop(){

    if(top==-1){

        printf("Stack Underflow\n");

    }

    else{

        printf("The element deleted is:%d\n",st[top]);

        top--;

    }

}

void peek(){

    if(top==-1){

        printf("Stack is Empty.\n");

    }

    else{

        printf("The element on top is:%d\n",st[top]);

    }

}

void display(){

    if(top==-1){

        printf("Stack is Empty.\n");

    }

    else{

        for(int i=0;i<=top;i++){

            printf("%d ",st[i]);

        }

        printf("\n");

    }

}

EXP 4

#include <stdio.h>

int f = -1, r = -1, q[100], size;

void insert() {

int val;

if (r == size - 1) {

printf("Overflow\n");

return;

}

printf("Enter value: ");

scanf("%d", &val);

q[++r] = val;

if (f == -1) f = 0;

}

void delete() {

if (f == -1) {

printf("Underflow\n");

return;

}

printf("Deleted: %d\n", q[f]);

if (f == r) f = r = -1;

else f++;

}

void display() {

if (f == -1) printf("Empty\n");

else for (int i = f; i <= r; i++) printf("%d ", q[i]);

printf("\n");

}

int main() {

printf("Enter size: ");

scanf("%d", &size);

int ch;

while (1) {

printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter choice: ");

scanf("%d", &ch);

switch (ch) {

case 1: insert(); break;

case 2: delete(); break;

case 3: display(); break;

case 4: printf("Exiting...\n"); return 0;

default: printf("Invalid choice\n");

}

}

}

EXP 5

#include <stdio.h>

#define MAX 100

int f = -1, r = -1, q[MAX], size;

void insert() {

int val;

if ((r + 1) % size == f) {

printf("Overflow\n");

return;

}

printf("Enter value: ");

scanf("%d", &val);

if (f == -1) f = r = 0;

else r = (r + 1) % size;

q[r] = val;

}

void delete() {

if (f == -1) {

printf("Underflow\n");

return;

}

printf("Deleted: %d\n", q[f]);

if (f == r) f = r = -1;

else f = (f + 1) % size;

}

void display() {

if (f == -1) {

printf("Empty\n");

return;

}

int i = f;

do {

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

i = (i + 1) % size;

} while (i != (r + 1) % size);

printf("\n");

}

int main() {

printf("Enter size: ");

scanf("%d", &size);

int ch;

while (1) {

printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");

printf("Enter choice: ");

scanf("%d", &ch);

switch (ch) {

case 1: insert(); break;

case 2: delete(); break;

case 3: display(); break;

case 4: return 0;

default: printf("Invalid choice\n");

}

}

}

EXP 6

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* insertAtBeginning(Node\* start, int value) {

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

newNode->data = value;

newNode->next = start;

return newNode;

}

Node\* insertAtEnd(Node\* start, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (!start) return newNode;

Node\* temp = start;

while (temp->next) temp = temp->next;

temp->next = newNode;

return start;

}

Node\* insertBefore(Node\* start, int value, int beforeValue) {

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

newNode->data = value;

if (!start) return start;

if (start->data == beforeValue) {

newNode->next = start;

return newNode;

}

Node\* temp = start;

while (temp->next && temp->next->data != beforeValue) temp = temp->next;

if (!temp->next) return start;

newNode->next = temp->next;

temp->next = newNode;

return start;

}

Node\* deleteNode(Node\* start, Node\* nodeToDelete) {

if (!start) return NULL;

if (start == nodeToDelete) {

start = start->next;

free(nodeToDelete);

return start;

}

Node\* temp = start;

while (temp->next && temp->next != nodeToDelete) temp = temp->next;

if (!temp->next) return start;

temp->next = temp->next->next;

free(nodeToDelete);

return start;

}

Node\* deleteAtBeginning(Node\* start) {

if (!start) return NULL;

return deleteNode(start, start);

}

Node\* deleteAtEnd(Node\* start) {

if (!start) return NULL;

if (!start->next) {

free(start);

return NULL;

}

Node\* temp = start;

while (temp->next->next) temp = temp->next;

return deleteNode(start, temp->next);

}

Node\* deleteValue(Node\* start, int value) {

if (!start) return NULL;

Node\* temp = start;

while (temp && temp->data != value) temp = temp->next;

if (!temp) return start;

return deleteNode(start, temp);

}

void display(Node\* start) {

printf("Linked List: ");

while (start) {

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

start = start->next;

}

printf("NULL\n");

}

int main() {

Node\* start = NULL;

int choice, value, beforeValue;

while (1) {

printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert Before\n4. Delete from Beginning\n5. Delete from End\n6. Delete Value\n7. Display\n8. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: printf("Enter value: "); scanf("%d", &value); start = insertAtBeginning(start, value); break;

case 2: printf("Enter value: "); scanf("%d", &value); start = insertAtEnd(start, value); break;

case 3: printf("Enter value: "); scanf("%d", &value); printf("Enter value before: "); scanf("%d", &beforeValue); start = insertBefore(start, value, beforeValue); break;

case 4: start = deleteAtBeginning(start); break;

case 5: start = deleteAtEnd(start); break;

case 6: printf("Enter value: "); scanf("%d", &value); start = deleteValue(start, value); break;

case 7: display(start); break;

case 8: return 0;

default: printf("Invalid choice.\n");

}

}

return 0;

}

EXP 7

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* top = NULL;

void push(int value) {

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

if (!newNode) {

printf("Stack Overflow!\n");

return;

}

newNode->data = value;

newNode->next = top;

top = newNode;

printf("%d pushed.\n", value);

}

void pop() {

if (!top) {

printf("Stack Underflow!\n");

return;

}

Node\* temp = top;

printf("%d popped.\n", top->data);

top = top->next;

free(temp);

}

void peek() {

if (top) printf("Top element: %d\n", top->data);

else printf("Stack is empty!\n");

}

void display() {

if (!top) {

printf("Stack is empty!\n");

return;

}

Node\* temp = top;

printf("Stack elements: ");

while (temp) {

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

temp = temp->next;

}

printf("\n");

}

int main() {

int choice, value;

while (1) {

printf("\n1. Push\n2. Pop\n3. Peek\n4. Display\n5. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: printf("Enter value: "); scanf("%d", &value); push(value); break;

case 2: pop(); break;

case 3: peek(); break;

case 4: display(); break;

case 5: printf("Exiting...\n"); return 0;

default: printf("Invalid choice!\n");

}

}

return 0;

}

EXP 8

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

typedef struct Queue {

Node \*front, \*rear;

} Queue;

Queue\* createQueue() {

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->front = q->rear = NULL;

return q;

}

int isEmpty(Queue\* q) {

return (q->front == NULL);

}

void enqueue(Queue\* q, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (isEmpty(q)) q->front = q->rear = newNode;

else q->rear->next = newNode, q->rear = newNode;

printf("%d enqueued.\n", value);

}

void dequeue(Queue\* q) {

if (isEmpty(q)) printf("Queue is empty! Cannot dequeue.\n");

else {

Node\* temp = q->front;

int data = temp->data;

q->front = q->front->next;

if (!q->front) q->rear = NULL;

free(temp);

printf("%d dequeued.\n", data);

}

}

void displayQueue(Queue\* q) {

if (isEmpty(q)) printf("Queue is empty!\n");

else {

Node\* temp = q->front;

printf("Queue elements: ");

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

printf("\n");

}

}

int main() {

Queue\* q = createQueue();

int choice, value;

while (1) {

printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: printf("Enter value: "); scanf("%d", &value); enqueue(q, value); break;

case 2: dequeue(q); break;

case 3: displayQueue(q); break;

case 4: printf("Exiting...\n"); return 0;

default: printf("Invalid choice!\n");

}

}

return 0;

}

EXP 9

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node \*left, \*right;

} Node;

Node\* createNode(int data) {

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

newNode->data = data;

newNode->left = newNode->right = NULL;

return newNode;

}

Node\* insertNode(Node\* root, int data) {

if (!root) return createNode(data);

if (data < root->data) root->left = insertNode(root->left, data);

else if (data > root->data) root->right = insertNode(root->right, data);

return root;

}

void inorder(Node\* root) {

if (root) {

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(Node\* root) {

if (root) {

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(Node\* root) {

if (root) {

postorder(root->left);

postorder(root->right);

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

}

}

int main() {

Node\* root = NULL;

int choice, value;

while (1) {

printf("\n1. Insert\n2. Inorder\n3. Preorder\n4. Postorder\n5. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value: ");

scanf("%d", &value);

root = insertNode(root, value);

break;

case 2: printf("Inorder: "); inorder(root); break;

case 3: printf("Preorder: "); preorder(root); break;

case 4: printf("Postorder: "); postorder(root); break;

case 5: exit(0);

default: printf("Invalid choice!\n");

}

printf("\n");

}

return 0;

}

EXP 10

#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#define MAX 100

int stack[MAX], top = -1;

void push(int x) {

if (top >= MAX - 1) exit(1);

stack[++top] = x;

}

int pop() {

if (top < 0) exit(1);

return stack[top--];

}

int evaluatePostfix(char \*exp) {

int op1, op2;

for (; \*exp; exp++) {

if (isspace(\*exp)) continue;

if (isdigit(\*exp)) push(\*exp - '0');

else {

op2 = pop(); op1 = pop();

switch (\*exp) {

case '+': push(op1 + op2); break;

case '-': push(op1 - op2); break;

case '\*': push(op1 \* op2); break;

case '/':

if (!op2) exit(1);

push(op1 / op2);

break;

default: exit(1);

}

}

}

return pop();

}

int main() {

char exp[MAX];

printf("Enter postfix expression: ");

fgets(exp, sizeof(exp), stdin);

printf("Result: %d\n", evaluatePostfix(exp));

return 0;

}

Best of luck

Gandu log