//WAP to create a linked list for stack and queue implentation

#include<stdio.h>

#include<stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void push(struct Node\*\* head\_ref, int new\_data) {

struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

new\_node->data = new\_data;

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

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

}

void Pop(struct Node\*\* head) {

struct Node \*ptr;

if (\*head == NULL) {

printf("\nList is empty");

} else {

ptr = \*head;

\*head = ptr->next;

free(ptr);

printf("Node deleted from the beginning\n");

}

}

void Enqueue(struct Node\*\* head, int item) {

struct Node \*ptr, \*temp;

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

ptr->data = item;

ptr->next = NULL;

if (\*head == NULL) {

\*head = ptr;

printf("\nNode inserted");

} else {

temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = ptr;

printf("\nNode inserted");

}

}

void Dequeue(struct Node \*\*head) {

struct Node \*ptr;

if (\*head == NULL) {

printf("\nList is empty");

} else {

ptr = \*head;

\*head = ptr->next;

free(ptr);

printf("Node deleted from the beginning\n");

}

}

void PrintStack(struct Node\* head) {

if (head == NULL) {

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

return;

}

printf("Stack: ");

while (head != NULL) {

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

head = head->next;

}

void PrintQueue(struct Node\* head) {

if (head == NULL) {

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

return;

}

printf("Queue: ");

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

}

printf("\n");

}

void ClearStack(struct Node\*\* head)

{

struct Node\* temp;

while (\*head != NULL) {

temp = \*head;

\*head = (\*head)->next;

free(temp);

}

printf("List is cleared\n");

}

int main()

{

struct Node \*head = NULL;

int ch, item;

do {

printf("\n1. Insert into stack");

printf("\n2. Delete from stack");

printf("\n3. Display stack");

printf("\n4. Insert into queue");

printf("\n5. Delete from queue");

printf("\n6. Display queue");

printf("\n7. Clear Linked list");

printf("\n8. Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch) {

case 1:

printf("\nEnter the value to be inserted: ");

scanf("%d", &item);

push(&head, item);

break;

case 2:

Pop(&head);

break;

case 3:

PrintStack(head);

break;

case 4:

printf("\nEnter the value to be inserted: ");

scanf("%d", &item);

Enqueue(&head, item);

break;

case 5:

Dequeue(&head);

break;

case 6:

PrintQueue(head);

break;

case 7:

ClearStack(&head);

break;

case 8:

return 0;

break;

default :

printf("Incorrect choice!");

}

} while (ch != 8);

return 0;

}



