1.Stack implementation using single linked list

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

}\*top,\*temp;

void push(int value);

int pop();

void display();

void main()

{

int choice,value;

printf("Implementation of stack using single linked list\n");

printf("1.push\t2.pop\t3.display\t4.exit\n");

while(1)

{

printf("Enter choice:\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter the value to be inserted:");

scanf("%d",&value);

push(value);

break;

case 2:printf("The popped value is %d\n",pop());

break;

case 3:display();

break;

case 4:exit(0);

break;

default:printf("Invalid input");

}

}

}

void push(int value)

{

if (top == NULL)

{

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

top->data = value;

top->next = NULL;

}

else

{

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

temp->data = value;

temp->next = top;

top = temp;

}

printf("Node inserted\n");

}

int pop()

{

temp=top;

if(top==NULL)

{

printf("Stack underflow");

return -1;

}

else{

temp=temp->next;

}

int popped=top->data;

free(top);

top=temp;

return popped;

}

void display()

{

temp = top;

printf("The elements in stack are\n");

if (top == NULL)

{

printf("Stack underflow\n");

return;

}

while (temp != NULL)

{

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

temp = temp->next;

}

printf("NULL\n");

}

2.Queue implementation using single linked list

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

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

void enqueue(int value);

int dequeue();

void display();

void main()

{

int choice,value;

printf("Implementation of queue using single linked list\n");

printf("\n1.enqueue\t2.dequeue\t3.display\t4.exit\n");

while(1)

{

printf("Enter choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter the value to be inserted:");

scanf("%d",&value);

enqueue(value);

break;

case 2:printf("The popped value is %d\n",dequeue());

break;

case 3:display();

break;

case 4:exit(0);

break;

default:printf("Invalid input");

}

}

}

void enqueue(int value)

{

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

ptr->data=value;

ptr->next=NULL;

if(front==NULL && rear==NULL)

{

front=rear=ptr;

}

else{

rear->next=ptr;

rear=ptr;

}

printf("Node inserted\n");

}

int dequeue()

{

if(front==NULL)

{

printf("Queue is empty");

return -1;

}

else{

struct node \*temp=front;

int temp\_data=front->data;

front=front->next;

free(temp);

return temp\_data;

}

}

void display()

{

if(front==NULL)

{

printf("Queue is empty");

}

struct node \*temp=front;

while(temp!=NULL)

{

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

temp=temp->next;

}

printf("NULL\n");

}

3.Sort and Reverse

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*start=NULL;

void create\_ll();

void sort\_ll();

void display();

void reverse();

void main()

{

printf("1.create linked list\n2.sort linked list\n3.display\n4.reverse the list\n5.exit");

int choice;

while(1)

{

printf("\nEnter choice");

scanf("%d",&choice);

switch(choice)

{

case 1:create\_ll();

break;

case 2:sort\_ll();

break;

case 3:display();

break;

case 4:reverse();

break;

case 5:exit(0);

break;

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

}

}

}

void create\_ll()

{

int option, num;

struct node \*ptr, \*newnode;

while (1)

{

printf("\nEnter 1. Creating list 2. Exit\n");

scanf("%d", &option);

switch (option)

{

case 1:

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

printf("Enter the value to be inserted: ");

scanf("%d", &num);

newnode->data = num;

newnode->next = NULL;

if (start == NULL)

{

start = newnode;

}

else

{

ptr = start;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = newnode;

}

break;

case 2:

return;

default:

printf("Invalid input\n");

}

}

}

void display()

{

if (start == NULL)

{

printf("\nList is empty\n");

return;

}

struct node \*ptr;

ptr=start;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

void sort\_ll()

{

struct node \*ptr1,\*ptr2;

int temp;

ptr1=start;

while(ptr1->next!=NULL)

{

ptr2=ptr1->next;

while(ptr2!=NULL)

{

if(ptr1->data > ptr2->data)

{

temp=ptr1->data;

ptr1->data=ptr2->data;

ptr2->data=temp;

}

ptr2=ptr2->next;

}

ptr1=ptr1->next;

}

}

void reverse()

{

struct node \*prev=NULL,\*temp=NULL,\*current=start;

while(current!=NULL)

{

temp=current->next;

current->next=prev;

prev=current;

current=temp;

}

start=prev;

}

4.Concatenate two lists

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head = NULL;

void display(struct node \*current)

{

if (current == NULL)

{

printf("NULL\n");

}

else

{

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

display(current->next);

}

}

void concatenate(struct node \*a, struct node \*b)

{

if (a != NULL && b != NULL)

{

if (a->next == NULL)

a->next = b;

else

concatenate(a->next, b);

}

else

{

printf("Either a or b is NULL\n");

}

}

int main()

{

struct node \*prev, \*a, \*b, \*p;

int n, i;

printf("Number of elements in a: ");

scanf("%d", &n);

printf("Enter the elements in a\n");

a = NULL;

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

{

p = malloc(sizeof(struct node));

scanf("%d", &p->data);

p->next = NULL;

if (a == NULL)

a = p;

else

prev->next = p;

prev = p;

}

printf("Number of elements in b: ");

scanf("%d", &n);

b = NULL;

printf("Enter the elements in b\n");

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

{

p = malloc(sizeof(struct node));

scanf("%d", &p->data);

p->next = NULL;

if (b == NULL)

b = p;

else

prev->next = p;

prev = p;

}

concatenate(a, b);

printf("Concatenated list: ");

display(a);

return 0;

}