#include <stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

void print(struct node\*ptr){

printf("printing elements of linked list");

printf(" ");

while(ptr!=NULL){

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

printf("->");

ptr= ptr->next;

}

}

struct node \* InsertAtHead(struct node\* first, int data){

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

temp->next = first;

temp->data = data;

first = temp;

return temp;

}

int main ()

{

struct node \*first;

struct node \*second;

struct node \*third;

struct node \*fourth;

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

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

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

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

first->data = 12;

first->next = second;

second->data = 13;

second->next = third;

third->data = 14;

third->next = fourth;

fourth->data = 15;

fourth->next = NULL;

print(first);

struct node head = (struct node) malloc (sizeof(struct node));

head =InsertAtHead( first,23 );

print(head);

}

#include <stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

void print(struct node\*ptr){

printf("printing elements of linked list");

printf(" ");

while(ptr!=NULL){

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

printf("->");

ptr= ptr->next;

}

}

struct node \* InsertAtHead(struct node\* first, int data){

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

temp->next = first;

temp->data = data;

first = temp;

return temp;

}

struct node \* InsertAtPos(struct node\* first, int data,int position){

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

struct node\*ptr = first;

while(position!=1){

ptr = ptr->next;

position--;

}

temp->next= ptr->next;

ptr->next= temp;

temp->data = data;

return first;

}

struct node\* InsertAtEnd(struct node\*first, int data){

struct node \* ptr = first;

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

while(ptr->next!= NULL){

ptr= ptr->next;

}

ptr->next = temp;

temp->next = NULL;

temp->data = data;

return first;

}

int main ()

{

struct node \*first;

struct node \*second;

struct node \*third;

struct node \*fourth;

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

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

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

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

first->data = 12;

first->next = second;

second->data = 13;

second->next = third;

third->data = 14;

third->next = fourth;

fourth->data = 15;

fourth->next = NULL;

print(first);

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

head =InsertAtHead( first,23 );

print(head);

head =InsertAtPos(first,56,3);

print(head);

head =InsertAtEnd(first,45);

print(head);

}

#include <stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

void print(struct node\*ptr){

printf("printing elements of linked list");

printf(" ");

while(ptr!=NULL){

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

printf("->");

ptr= ptr->next;

}

}

struct node \* InsertAtHead(struct node\* first, int data){

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

temp->next = first;

temp->data = data;

first = temp;

return temp;

}

struct node \* InsertAtPos(struct node\* first, int data,int position){

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

struct node\*ptr = first;

while(position!=1){

ptr = ptr->next;

position--;

}

temp->next= ptr->next;

ptr->next= temp;

temp->data = data;

return first;

}

struct node\* InsertAtEnd(struct node\*first, int data){

struct node \* ptr = first;

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

while(ptr->next!= NULL){

ptr= ptr->next;

}

ptr->next = temp;

temp->next = NULL;

temp->data = data;

return first;

}

struct node \*Deletefromhead(struct node\*first){

struct node\* temp = first;

first= first->next;

temp->next = NULL;

free(temp);

return first;

}

struct node \*DeleteFromPosition(struct node\*first,int position){

struct node\* prevnode= NULL;

struct node\* currnode = first;

while(position!=1){

prevnode = currnode;

currnode = currnode->next;

position--;

}

prevnode->next = currnode->next;

currnode->next = NULL;

free(currnode);

return first;

}

struct node\* DeleteFromEnd(struct node\*first){

struct node\* prevnode = NULL;

struct node\* nextnode =first;

while(nextnode->next!=NULL){

prevnode = nextnode;

nextnode=prevnode->next;

}

prevnode->next =NULL;

free(nextnode);

return first;

}

int main ()

{

struct node \*first;

struct node \*second;

struct node \*third;

struct node \*fourth;

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

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

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

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

first->data = 12;

first->next = second;

second->data = 13;

second->next = third;

third->data = 14;

third->next = fourth;

fourth->data = 15;

fourth->next = NULL;

print(first);

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

print(first);

head = DeleteFromPosition(first, 2);

print(head);

head = DeleteFromEnd(first);

print(head);

}

#include <stdio.h>

#include<stdlib.h>

struct node {

int data;

struct node\*prev;

struct node \*next;

};

void print(struct node\*first){

struct node \*temp = first;

while(temp!=NULL){

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

temp= temp->next;

}

}

struct node\*InsertAtHead(struct node\*first,int data){

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

ptr->next= first;

ptr->data= data;

first->prev =ptr;

ptr->prev =NULL;

first = ptr;

return first;

}

int main() {

struct node\* first;

struct node\* second;

struct node\* third;

struct node\* fourth;

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

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

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

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

first->prev = NULL;

first->data = 12;

first->next=second;

second->prev = first;

second->data= 13;

second->next = third;

third->prev = second;

third->data= 14;

third->next = fourth;

fourth->prev = third;

fourth->data= 15;

fourth->next = first;

print(first);

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

head =InsertAtHead(first,2);

print(head);

}

#include <stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

void print(struct node\*ptr){

while(ptr!=NULL){

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

printf("->");

ptr= ptr->next;

}

}

struct node\* ReverseLinkedlist(struct node\* first){

struct node\* prevnode = NULL;

struct node\* currnode = first;

while(currnode!=NULL){

struct node\* nextnode =currnode->next ;

currnode->next=prevnode;

prevnode = currnode;

currnode= nextnode;

}

return prevnode;

}

int main ()

{

struct node \*first;

struct node \*second;

struct node \*third;

struct node \*fourth;

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

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

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

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

first->data = 12;

first->next = second;

second->data = 13;

second->next = third;

third->data = 14;

third->next = fourth;

fourth->data = 15;

fourth->next = NULL;

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

head =ReverseLinkedlist(first);

print(head);

}

#include <stdio.h>

#include <stdlib.h>

int stack[5];

int top =-1;

void push(int data){

if(top==4){

printf("stack overflow");

}

else{

top++;

stack[top]=data;

}

}

void pop(){

if(top==-1){

printf("stack underflow");

}

else{

top--;

}

}

void peek(){

printf("%d",stack[top]);

}

void display(){

for(int i = top; i>=0; i--){

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

}

}

int main() {

// Write C code here

push(5);

push(4);

push(6);

push(8);

push(9);

display();

pop();

display();

}

#include <stdio.h>

#include <stdlib.h>

// A linked list (LL) node to store a queue entry

struct node {

int data;

struct node\* next;

};

struct node\* front = NULL;

struct node\* rear = NULL;

void enqueue(int val){

struct node\* newnode;

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

if(newnode==NULL){

printf("Queue over flow");

}

if(front==NULL){

newnode->data= val;

newnode->next=NULL;

front = newnode;

rear= newnode;

}

else{

rear->next = newnode;

newnode->data= val;

rear = newnode;

}

}

void dequeue(){

if(front==NULL){

printf("underflow");

}

else{

front->data = -1;

front = front->next;

}

}

void print(struct node\*ptr){

while(ptr!=NULL){

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

ptr= ptr->next;

}

}

int main(){

enqueue(5);

enqueue(9);

enqueue(4);

dequeue();

dequeue();

dequeue();

dequeue();

print(front);

}

#include <stdio.h>

#include<stdlib.h>

struct node {

int data;

struct node\*prev;

struct node \*next;

};

int temp;

void print(struct node\*first){

struct node \*temp = first;

while(temp!=NULL){

printf("ELEMENTS ARE: %d\n" ,temp->data);

temp= temp->next;

}

}

struct node\*InsertAtHead(struct node\*first,int data){

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

ptr->next= first;

ptr->data= data;

first->prev =ptr;

ptr->prev =NULL;

first = ptr;

return first;

}

void Sort(struct node\*first){

struct node\* curr=NULL;

struct node\* index=NULL;

int temp;

for(curr=first; curr->next!=NULL;curr= curr->next){

for(index = curr->next;index!=NULL; index = index->next){

if(curr->data > index->data){

temp = curr->data;

curr->data = index->data;

index->data= temp;

}

}

}

}

int main() {

struct node\* first;

struct node\* second;

struct node\* third;

struct node\* fourth;

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

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

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

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

first->prev = NULL;

first->data = 12;

first->next=second;

second->prev = first;

second->data= 14;

second->next = third;

third->prev = second;

third->data= 13;

third->next = fourth;

fourth->prev = third;

fourth->data= 11;

fourth->next = NULL;

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

Sort(first);

print(first);

}