1. **Write a program to construct a queue using Linked List with comments on each line.**

#include <stdio.h>

#include <stdlib.h>

struct node{ //structure for node

int info;

struct node \*next;

};

typedef struct node NodeType; //initiallizing user define datatpe

NodeType \*rear=0,\*front=0; //initillizing value of rear and front

void insert(int);

void display();

int main(){

int choice, item;

do

{

printf("\n1.Insert \n2.Display\n3:Exit\n");

printf("enter ur choice\n");

scanf("%d",&choice);

switch(choice){

case 1:

printf("\nEnter the data:\n");

scanf("%d",&item);

insert(item);

break;

case 2:

display();

break;

//case 3:

// exit(1);

// break;

default:

printf("invalid choice\n");

break;

}

}while(choice<3);

return 0;

}

//insert data

void insert(int item) {

NodeType \*nnode;

nnode=( NodeType \*)malloc(sizeof(NodeType));

if(rear==0) { //condotion where there is no data inseated yet

nnode->info=item; //inseating the value in nnode

nnode->next=NULL; //initilizing null value at nnode->next

rear=front=nnode; //changing the position of rear and front fron null to nnode

}

else {//condition where there is already some items inseated in queue

nnode->info=item; //insearting the item in queue

nnode->next=NULL; //initilizing the nnode-> to null so more data can be inserted later

rear->next=nnode; //replacing the value of rear->next with address of nnode so new node points to previous node

rear=nnode; //changing rear to new node

}

}

//display function

void display() {

NodeType \*temp; //initilizing temp variable

temp=front; // replacing the vall=ue of tem with front

printf("\nqueue items are:\t");

while(temp!=NULL){ //loop which display all items of queue until temp reaches to last item i.e null

printf("%d\t",temp->info);

temp=temp->next;

}

}

**2. Write a program to construct a Circular Linked List with comments on each line.**

**3. Write a program to implement Stack as a circular list with comments on each line.**

**4. Write a program to implement queue as a circular list with comments on each line.**

**5. Write a program to implement Doubly Linked List with comments on each line.**

/\* Doubly Linked List implementation \*/

#include<stdio.h>

#include<stdlib.h>

struct Node { //creating structure for node

int data;

struct Node\* next;

struct Node\* prev;

};

struct Node\* head; // global variable - pointer to head node.

//Creates a new Node and returns pointer to it.

struct Node\* GetNewNode(int x) { //creating newnode

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

newNode->data = x;

newNode->prev = NULL;

newNode->next = NULL;

return newNode;

}

//Inserts a Node at head of doubly linked list

void InsertAtHead(int x) {

struct Node\* newNode = GetNewNode(x);

if(head == NULL) {

head = newNode;

return;

}

head->prev = newNode;

newNode->next = head;

head = newNode;

}

//Inserts a Node at tail of Doubly linked list

void InsertAtTail(int x) {

struct Node\* temp = head;

struct Node\* newNode = GetNewNode(x);

if(head == NULL) {

head = newNode;

return;

}

while(temp->next != NULL) temp = temp->next; // Go To last Node

temp->next = newNode;

newNode->prev = temp;

}

//Prints all the elements in linked list in forward traversal order

void Print() {

struct Node\* temp = head;

printf("Forward: ");

while(temp != NULL) {

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

temp = temp->next;

}

printf("\n");

}

int main() {

head = NULL; // empty list. set head as NULL.

int choice, item;

do {

printf("\n1.insert from head \n2.inseart from tail \n3.Display\n4:Exit\n");

printf("enter ur choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the data:\n"); scanf("%d",&item); InsertAtHead(item);

break;

case 2:

printf("\nEnter the data:\n"); scanf("%d",&item); InsertAtTail(item);

break;

break;

case 3:

print();

break;

//case 4:

//exit(1);

//break;

default:

printf("invalid choice\n");

break;

}

}while(choice<5);

return 0;

}

**6. Write a program to implement Circular Doubly Linked List with comments on each line.**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int val;

struct node \*next;

struct node \*prev;

};

typedef struct node n;

n\* create\_node(int);

void add\_node();

void insert\_at\_first();

void insert\_at\_end();

void delete\_node\_position();

void display\_from\_beg();

void display\_in\_rev();

n \*new, \*ptr, \*prev;

n \*first = NULL, \*last = NULL;

nt number = 0;

//MEMORY ALLOCATED FOR NODE DYNAMICALLY

n\* create\_node(int info)

{

number++;

new = (n \*)malloc(sizeof(n));

new->val = info;

new->next = NULL;

new->prev = NULL;

return new;

}

/\*

\*ADDS NEW NODE

\*/

void add\_node()

{

int info;

printf("\nenter the value you would like to add:");

scanf("%d", &info);

new = create\_node(info);

if (first == last && first == NULL)

{

first = last = new;

first->next = last->next = NULL;

first->prev = last->prev = NULL;

}

else

{

last->next = new;

new->prev = last;

last = new;

last->next = first;

first->prev = last;

}

}

/\*

\*INSERTS ELEMENT AT FIRST

\*/

void insert\_at\_first()

{

int info;

printf("\nenter the value to be inserted at first:");

scanf("%d",&info);

new = create\_node(info);

if (first == last && first == NULL)

{

printf("\ninitially it is empty linked list later insertion is done");

first = last = new;

first->next = last->next = NULL;

first->prev = last->prev = NULL;

}

else

{

new->next = first;

first->prev = new;

first = new;

first->prev = last;

last->next = first;

printf("\n the value is inserted at begining");

}

}

/\*

\*INSERTS ELEMNET AT END

\*/

void insert\_at\_end()

{

int info;

printf("\nenter the value that has to be inserted at last:");

scanf("%d", &info);

new = create\_node(info);

if (first == last && first == NULL)

{

printf("\ninitially the list is empty and now new node is inserted but at first");

first = last = new;

first->next = last->next = NULL;

first->prev = last->prev = NULL;

}

else

{

last->next = new;

new->prev = last;

last = new;

first->prev = last;

last->next = first;

}

}

/\*

\*DELETION IS DONE

\*/

void delete\_node\_position()

{

int pos, count = 0, i;

n \*temp, \*prevnode;

printf("\n enter the position which u wanted to delete:");

scanf("%d", &pos);

if (first == last && first == NULL)

printf("\n empty linked list you cant delete");

else

{

if (number < pos)

printf("\n node cant be deleted at position as it is exceeding the linkedlist length");

else

{

for (ptr = first,i = 1;i <= number;i++)

{

prevnode = ptr;

ptr = ptr->next;

if (pos == 1)

{

number--;

last->next = prevnode->next;

ptr->prev = prevnode->prev;

first = ptr;

printf("%d is deleted", prevnode->val);

free(prevnode);

break;

}

else if (i == pos - 1)

{

number--;

prevnode->next = ptr->next;

ptr->next->prev = prevnode;

printf("%d is deleted", ptr->val);

free(ptr);

break;

}

}

}

}

}

/\*

\*UPDATION IS DONE FRO GIVEN OLD VAL

\*/

if (ptr->val == key)

{

printf("\n the value is found at position at %d", count);

f = 1;

}

}

if (f == 0)

printf("\n the value is not found in linkedlist");

}

}

/\*

\*DISPLAYING IN BEGINNING

\*/

void display\_from\_beg()

{

int i;

if (first == last && first == NULL)

printf("\nlist is empty no elemnts to print");

else

{

printf("\n%d number of nodes are there", number);

for (ptr = first, i = 0;i < number;i++,ptr = ptr->next)

printf("\n %d", ptr->val);

}

}

/\*

\* DISPLAYING IN REVERSE

\*/

void display\_in\_rev()

{

int i;

if (first == last && first == NULL)

printf("\nlist is empty there are no elments");

else

{

for (ptr = last, i = 0;i < number;i++,ptr = ptr->prev)

{

printf("\n%d", ptr->val);

}

}