**Q1 Major: - W.A. Program for Create 5 nodes and display Linked List for Single Linked List. [**

**Minor:- Insert an element at 4th place**

#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node \*link;

}\*start=NULL,\*q;

void createnode(int data);

void display();

void insertatposition(int data,int pos);

int main(){

int i,ch,n,m,p;

while (1)

{

printf("Create a list\n");

printf("Display a list\n");

printf("Insert a node\n");

printf("Enter a choice");

scanf("%d",&ch);

switch(ch){

case 1:

printf("How many nodes you want enter");

scanf("%d",&n);

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

{

printf("Enter the data %d node",i+1);

scanf("%d",&m);

createnode(m);

}

break;

case 2:

display();

break;

case 3:

printf("Enter the position");

scanf("%d",&p);

printf("Enter the data");

scanf("%d",&m);

insertatposition(m,p);

break;

default:

printf("Wrong choice");

break;

}

}

}

void createnode(int data){

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

tmp->info=data;

tmp->link=NULL;

if(start==NULL){

start=tmp;

}else{

q=start;

while (q->link!=NULL)

{

q=q->link;

}

q->link=tmp;

}

}

void display(){

q=start;

if(start==NULL){

printf("LIST IS EMPTY.\n");

return;

}else{

printf("%d",q->info);

while(q->link!=NULL){

q=q->link;

printf("%d",q->info);

}

}

}

void insertatposition(int data,int pos){

int i;

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

tmp->info=data;

if (pos==1)

{

tmp->link=start;

start=tmp;

return;

}

q=start;

for(i=1;i<pos-1&&q!=NULL;i++){

q=q->link;

}

if(q==NULL){

printf("There are less than &d elements",pos);

}else{

tmp->link=q->link;

q->link=tmp;

}

}

**Q2.Major: - W.A. Program for Create 10 nodes and display Linked List for Circular Linked List.**

**Minor:- Insert an element at First place.**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

}\*last=NULL,\*q;

void createnode(int data);

void display();

void insertatbeg(int data);

void main(){

int i,ch,n,m,p;

while(1){

printf("Create a list\n");

printf("display\n");

printf("Insert Node\n");

printf("Enter your choice\n");

scanf("%d",&ch);

switch(ch){

case 1:

printf("How many node do you want to enter");

scanf("%d",&n);

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

printf("enter the data for %d node",i+1);

scanf("%d",&m);

createnode(m);

}

break;

case 2:

display();

break;

case 3:

printf("Enter the data\n");

scanf("%d",&m);

insertatbeg(m);

}

}

}

void createnode(int data){

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

tmp->info=data;

if(last==NULL){

last=tmp;

tmp->link=last;

}

tmp->link=last->link;

last->link=tmp;

last=tmp;

}

void display(){

q=last->link;

do{

printf("%d",q->info);

q=q->link;

}while(q!=last->link);

}

void insertatbeg(int data){

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

tmp->info=data;

tmp->link=last->link;

last->link=tmp;

}

**Q3.Major: - W.A. Program for Create 5 nodes and display Linked List for double Linked List.**

**Minor:- Ask the key value form user for searching and display the element position index number.**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*prev;

struct node \*next;

}\*start=NULL,\*q;

void createnode(int data);

void display();

void search(int data);

int main(){

int n,m,ch;

while(1){

printf("create node\n");

printf("Disply List\n");

printf("Search\n");

printf("enter your choice");

scanf("%d",&ch);

switch (ch)

{

case 1:

printf("how many you want to enter ");

scanf("%d",&n);

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

printf("enter the data for %d node",i+1);

scanf("%d",&m);

createnode(m);

}

break;

case 2:

display();

break;

case 3:

printf("Enter the search data");

scanf("%d",&m);

search(m);

}

}

}

void createnode(int data){

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

tmp->info=data;

tmp->next=NULL;

tmp->prev=NULL;

if(start==NULL){

start=tmp;

}else{

q=start;

while(q->next!=NULL){

q=q->next;

}

q->next=tmp;

tmp->prev=q;

}

}

void display(){

if(start==NULL){

printf("list is empty");

}

start=q;

while(q!=NULL){

printf("%d",q->info);

q=q->next;

}

}

void search(int data){

q=start;

int pos=1;

while(q!=NULL){

if(q->info==data){

printf("Data %d found at postion %d.\n",data,pos);

return;

}

q=q->next;

pos++;

}

printf("Data %d not found in the list\n",data);

}

struct node

}

**Q.4Major: - W.A. Program for Create 5 nodes and display Linked List for Single Linked List**

**Minor:- display in reverse created linked list**

#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node \*link;

}\*start=NULL,\*q;

void createnode(int data);

void display();

void reverse();

int main(){

int i,ch,n,m,p;

while (1)

{

printf("Create a list\n");

printf("Display a list\n");

printf("Reverse\n");

printf("Enter your choise");

scanf("%d",&ch);

switch(ch){

case 1:

printf("How many nodes you want enter");

scanf("%d",&n);

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

{

printf("Enter the data %d node",i+1);

scanf("%d",&m);

createnode(m);

}

break;

case 2:

display();

break;

case 3:

reverse();

break;

default:

printf("Wrong choice");

break;

}

}

}

void createnode(int data){

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

tmp->info=data;

tmp->link=NULL;

if(start==NULL){

start=tmp;

}else{

q=start;

while (q->link!=NULL)

{

q=q->link;

}

q->link=tmp;

}

}

void display(){

q=start;

if(start==NULL){

printf("LIST IS EMPTY.\n");

return;

}else{

printf("%d",q->info);

while(q->link!=NULL){

q=q->link;

printf("%d",q->info);

}

}

}

void reverse()

{

struct node \*p1,\*p2,\*p3;

if(start->link==NULL){

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

}else{

p1=start;

p2=p1->link;

p3=p2->link;

p1->link=NULL;

p2->link=p1;

while(p3!=NULL){

p1=p2;

p2=p3;

p3=p2->link;

p2->link=p1;

}

start=p2;

}

q=start;

while (q!=NULL)

{

printf("\n %d",q->info);

q=q->link;

}

}

**Q.5 Major: - W.A. Program for Push 4 elements and display stack by using Linked List**

**Minor:- 1. Insert an element on top place.**

**2. display the output “Nodes are not available”**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Stack {

struct Node\* top;

};

void initializeStack(struct Stack\* stack) {

stack->top = NULL;

}

void push(struct Stack\* stack, int data) {

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

if (newNode == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

newNode->data = data;

newNode->next = stack->top;

stack->top = newNode;

}

int pop(struct Stack\* stack) {

if (stack->top == NULL) {

return -1;

}

int data = stack->top->data;

struct Node\* temp = stack->top;

stack->top = stack->top->next;

free(temp);

return data;

}

void displayStack(struct Stack\* stack) {

if (stack->top == NULL) {

printf("Nodes are not available\n");

return;

}

struct Node\* current = stack->top;

while (current != NULL) {

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

current = current->next;

}

printf("\n");

}

int main() {

struct Stack myStack;

initializeStack(&myStack);

int choice, data;

do {

printf("\nMenu:\n");

printf("1. Insert an element on top\n");

printf("2. Display the stack and perform pop operations\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to insert: ");

scanf("%d", &data);

push(&myStack, data);

printf("Element %d inserted on top\n", data);

break;

case 2:

printf("Stack elements: ");

displayStack(&myStack);

printf("\nPop operations:\n");

for (int i = 0; i < 5; ++i) {

int poppedElement = pop(&myStack);

if (poppedElement != -1) {

printf("Popped: %d\n", poppedElement);

} else {

printf("Nodes are not available\n");

}

}

break;

case 3:

printf("Exiting program\n");

break;

default:

printf("Invalid choice\n");

}

} while (choice != 3);

return 0;

}

**Q.6Major: - W.A. Program for make a queue of 10 elements and display Queue by using Linked List.**

**Minor:- Delete 7th element from queue.**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Queue {

struct Node\* front;

struct Node\* rear;

};

void initializeQueue(struct Queue\* queue) {

queue->front = queue->rear = NULL;

}

void enqueue(struct Queue\* queue, int data) {

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

if (newNode == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

if (queue->rear == NULL) {

queue->front = queue->rear = newNode;

} else {

queue->rear->next = newNode;

queue->rear = newNode;

}

}

int dequeue(struct Queue\* queue) {

if (queue->front == NULL) {

return -1; // Queue is empty

}

int data = queue->front->data;

struct Node\* temp = queue->front;

queue->front = queue->front->next;

free(temp);

if (queue->front == NULL) {

queue->rear = NULL;

}

return data;

}

void displayQueue(struct Queue\* queue) {

if (queue->front == NULL) {

printf("Queue is empty\n");

return;

}

struct Node\* current = queue->front;

while (current != NULL) {

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

current = current->next;

}

printf("\n");

}

void delete7thElement(struct Queue\* queue) {

struct Node\* current = queue->front;

for (int i = 1; i < 7 && current != NULL; ++i) {

current = current->next;

}

if (current == NULL || current->next == NULL) {

printf("Unable to delete the 7th element. Insufficient elements in the queue.\n");

return;

}

struct Node\* temp = current->next;

current->next = temp->next;

free(temp);

}

int main() {

struct Queue myQueue;

initializeQueue(&myQueue);

for (int i = 1; i <= 10; ++i) {

enqueue(&myQueue, i \* 10);

}

printf("Original Queue elements: ");

displayQueue(&myQueue);

delete7thElement(&myQueue);

printf("\nUpdated Queue elements after manually deleting 7th element: ");

displayQueue(&myQueue);

return 0;

}

**Q.7- W.A. Program for Create 5 nodes and display Linked List for double Linked List in sorted order**

**while creating a list duplicate should be allowed.**

#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\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;

if (first != NULL) {

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;

}

}

}

}

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

if (data < 0) {

first = InsertAtHead(first, data);

} else {

printf("Cannot insert non-negative element.\n");

}

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 = 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 = NULL;

printf("Original List:\n");

print(first);

Sort(first);

printf("\nSorted List:\n");

print(first);

head = InsertNegative(first, -5);

printf("\nList after inserting a negative element:\n");

print(head);

return 0;

}