**Task 2.a Singly Linked List: Operations such as insertion, deletion and sorting in Ascending Order**

#include <stdio.h>

#include <malloc.h>

#define ISEMPTY printf("\nEMPTY LIST:");

struct node

{

int value;

struct node \*next;

};

typedef struct node snode;

snode\* create\_node(int);

void insert\_node\_first();

void insert\_node\_last();

void insert\_node\_pos();

void delete\_pos();

void display();

snode \*newnode, \*ptr, \*prev, \*temp;

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

int main() { int ch; char ans = 'Y';

while (ans == 'Y'||ans == 'y')

{

printf("\n---------------------------------\n");

printf("\nOperations on singly linked list\n");

printf("\n---------------------------------\n");

printf("\n1.Insert node at first");

printf("\n2.Insert node at last");

printf("\n3.Insert node at position");

printf("\n4.Delete Node from any Position");

printf("\n5.Display List from Beginning to end");

printf("\n6.Exit\n");

printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

printf("\nEnter your choice");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\n...Inserting node at first...\n");

insert\_node\_first(); break;

case 2:

printf("\n...Inserting node at last...\n");

insert\_node\_last();

break;

case 3:

printf("\n...Inserting node at position...\n");

insert\_node\_pos();

break;

case 4:

printf("\n...Deleting Node from any Position...\n");

delete\_pos();

break;

case 5:

printf("\n...Displaying List From Beginning to End...\n");

display(); break;

case 6:

printf("\n...Exiting...\n");

return 0;

break;

default:

printf("\n...Invalid Choice...\n");

break;

}

printf("\nYOU WANT TO CONTINUE (Y/N)");

scanf(" %c", &ans);

}

return 0;

}

snode\* create\_node(int val)

{

newnode = (snode \*)malloc(sizeof(snode));

if (newnode == NULL)

{

printf("\nMemory was not allocated");

return 0;

}

else

{

newnode->value = val;

newnode->next = NULL;

return newnode;

}

}

void insert\_node\_first()

{

int val;

printf("\nEnter the value for the node:");

scanf("%d", &val);

newnode = create\_node(val);

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

{

first = last = newnode;

first->next = NULL;

last->next = NULL;

}

else

{

temp = first;

first = newnode;

first->next = temp;

}

printf("\n----INSERTED----");

}

void insert\_node\_last()

{

int val;

printf("\nEnter the value for the Node:");

scanf("%d", &val);

newnode = create\_node(val);

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

{

first = last = newnode;

first->next = NULL;

last->next = NULL;

}

else

{

last->next = newnode;

last = newnode;

last->next = NULL;

}

printf("\n----INSERTED----");

}

void insert\_node\_pos()

{

int pos,

val, cnt = 0, i;

printf("\nEnter the value for the Node:");

scanf("%d", &val);

newnode = create\_node(val);

printf("\nEnter the position ");

scanf("%d", &pos);

ptr = first;

while (ptr != NULL)

{

ptr = ptr->next;

cnt++;

}

if (pos == 1)

{

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

{

first = last = newnode;

first->next = NULL;

last->next = NULL;

}

else

{

temp = first;

first = newnode;

first->next = temp;

}

printf("\nInserted");

}

else if (pos>1 && pos<=cnt)

{

ptr = first;

for (i = 1;i < pos;i++)

{

prev = ptr;

ptr = ptr->next;

}

prev->next = newnode;

newnode->next = ptr;

printf("\n----INSERTED----");

}

else

{

printf("Position is out of range");

}

}

void delete\_pos()

{

int pos, cnt = 0, i;

if (first == NULL)

{

ISEMPTY;

printf(":No node to delete\n");

}

else

{

printf("\nEnter the position of value to be deleted:");

scanf(" %d", &pos);

ptr = first; if (pos == 1)

{

first = ptr->next;

printf("\nElement deleted");

}

else

{

while (ptr != NULL)

{

ptr = ptr->next; cnt = cnt + 1;

}

if (pos > 0 && pos <= cnt)

{

ptr = first;

for (i = 1;i < pos;i++)

{

prev = ptr; ptr = ptr->next;

}

prev->next = ptr->next;

}

else

{

printf("Position is out of range");

}

free(ptr);

printf("\nElement deleted");

}

}

}

void display()

{

if (first == NULL)

{

ISEMPTY;

printf(":No nodes in the list to display\n");

}

else

{

for (ptr = first;ptr != NULL;ptr = ptr->next)

{

printf("%d\t", ptr->value); }

}

}

**Add this function in single linked list program**

void sorted\_ascend()

{

snode \*nxt;

int t;

if (first == NULL)

{

ISEMPTY;

printf(":No elements to sort\n");

}

else

{

for (ptr = first;ptr != NULL;ptr = ptr->next)

{

for (nxt = ptr->next;nxt != NULL;nxt = nxt->next)

{

if (ptr->value > nxt->value)

{

t = ptr->value;

ptr->value = nxt->value;

nxt->value = t;

}

}

}

printf("\n---Sorted List---");

for (ptr = first;ptr != NULL;ptr = ptr->next)

{

printf("%d\t", ptr->value);

}

}

}

**Task 2. B Doubly Linked List – Operations : insertion, deletion and display in reverse order**

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

struct node \*next;

int data;

};

struct node \*head;

void insertion\_beginning();

void insertion\_last();

void insertion\_specified();

void deletion\_beginning();

void deletion\_last();

void deletion\_specified();

void display();

void search();

void main ()

{

int choice =0;

while(choice != 9)

{

printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");

printf("\nChoose one option from the following list ...\n");

printf("\n===============================================\n");

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete the node after the given data\n5.Show\n6.Exit\n");

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

insertion\_beginning();

break;

case 2:

insertion\_last();

break;

case 3:

insertion\_specified();

break;

case 4:

deletion\_specified();

break;

case 5:

display();

break;

case 6:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void insertion\_beginning()

{

struct node \*ptr;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter Item value");

scanf("%d",&item);

if(head==NULL)

{

ptr->next = NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->prev=NULL;

ptr->next = head;

head->prev=ptr;

head=ptr;

}

printf("\nNode inserted\n");

}

}

void insertion\_last()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value");

scanf("%d",&item);

ptr->data=item;

if(head == NULL)

{

ptr->next = NULL;

ptr->prev = NULL;

head = ptr;

}

else

{

temp = head;

while(temp->next!=NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr ->prev=temp;

ptr->next = NULL;

}

}

printf("\nnode inserted\n");

}

void insertion\_specified()

{

struct node \*ptr,\*temp;

int item,loc,i;

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

if(ptr == NULL)

{

printf("\n OVERFLOW");

}

else

{

temp=head;

printf("Enter the location");

scanf("%d",&loc);

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

{

temp = temp->next;

if(temp == NULL)

{

printf("\n There are less than %d elements", loc);

return;

}

}

printf("Enter value");

scanf("%d",&item);

ptr->data = item;

ptr->next = temp->next;

ptr -> prev = temp;

temp->next = ptr;

temp->next->prev=ptr;

printf("\nnode inserted\n");

}

}

void deletion\_specified()

{

struct node \*ptr, \*temp;

int val;

printf("\n Enter the data after which the node is to be deleted : ");

scanf("%d", &val);

ptr = head;

while(ptr -> data != val)

ptr = ptr -> next;

if(ptr -> next == NULL)

{

printf("\nCan't delete\n");

}

else if(ptr -> next -> next == NULL)

{

ptr ->next = NULL;

}

else

{

temp = ptr -> next;

ptr -> next = temp -> next;

temp -> next -> prev = ptr;

free(temp);

printf("\nnode deleted\n");

}

}

void display()

{

struct node \*ptr;

printf("\n printing values...\n");

ptr = head;

while(ptr != NULL)

{

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

ptr=ptr->next;

}

}

**Create and Display a Doubly Linked List in Reverse Order**

void displayDlListRev()

{

struct node \* tmp;

int n = 0;

if(ennode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = ennode;

printf("\n Data in reverse order are :\n");

while(tmp != NULL)

{

printf(" Data in node %d : %d\n", n+1, tmp->num);

n++;

tmp = tmp->preptr; // current pointer set with previous node

}

}

}

**Task 2.c Cirular Linked List - Choose the place in the end of the circle so that you are**

**the last one.**

#include <stdio.h>

#include <stdlib.h>

struct node {

int num;

struct node \* nextptr;

}\*stnode;

void ClListcreation(int n);

void displayClList();

int main()

{

int n;

stnode = NULL;

printf("\n\n Circular Linked List : Create and display a circular linked list :\n");

printf("-----------------------------------------------------------------------\n");

printf(" Input the number of nodes : ");

scanf("%d", &n);

ClListcreation(n);

displayClList();

return 0;

}

void ClListcreation(int n)

{

int i, num;

struct node \*preptr, \*newnode;

if(n >= 1)

{

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

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->nextptr = NULL;

preptr = stnode;

for(i=2; i<=n; i++)

{

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

printf(" Input data for node %d : ", i);

scanf("%d", &num);

newnode->num = num;

newnode->nextptr = NULL; // next address of new node set as NULL

preptr->nextptr = newnode; // previous node is linking with new node

preptr = newnode; // previous node is advanced

}

preptr->nextptr = stnode; //last node is linking with first node

}

}

void displayClList()

{

struct node \*tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

printf("\n\n Data entered in the list are :\n");

printf(" Address of first node is %d\n", tmp);

do {

printf(" Data part of %d is %d\n", n, tmp->num);

printf(" Address part of %d is %d\n", n, tmp->nextptr);

tmp = tmp->nextptr;

n++;

}while(tmp != stnode);

}

}