**PF LAB ASSIGNMENT**

**13DEC2020**

**20k-0157**

**Task1**

/\* C Program to Concatenate two singly linked lists\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

};

struct node \*create\_list(struct node \*);

struct node \*concat( struct node \*start1,struct node \*start2);

struct node \*addatbeg(struct node \*start, int data);

struct node \*addatend(struct node \*start,int data);

void display(struct node \*start);

int main()

{

struct node \*start1=NULL,\*start2=NULL;

start1=create\_list(start1);

start2=create\_list(start2);

printf("\nFirst list is : ");

display(start1);

printf("\nSecond list is : ");

display(start2);

start1=concat(start1, start2);

printf("\nConcatenated list is : ");

display(start1);

return 0;

}

struct node \*concat( struct node \*start1,struct node \*start2)

{

struct node \*ptr;

if(start1==NULL)

{

start1=start2;

return start1;

}

if(start2==NULL)

return start1;

ptr=start1;

while(ptr->link!=NULL)

ptr=ptr->link;

ptr->link=start2;

return start1;

}

struct node \*create\_list(struct node \*start)

{

int i,n,data;

printf("\nEnter the number of nodes : ");

scanf("%d",&n);

start=NULL;

if(n==0)

return start;

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

scanf("%d",&data);

start=addatbeg(start,data);

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

{

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

scanf("%d",&data);

start=addatend(start,data);

}

return start;

}

void display(struct node \*start)

{

struct node \*p;

if(start==NULL)

{

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

return;

}

p=start;

while(p!=NULL)

{

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

p=p->link;

}

printf("\n");

}/\*End of display() \*/

struct node \*addatbeg(struct node \*start,int data)

{

struct node \*tmp;

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

tmp->info=data;

tmp->link=start;

start=tmp;

return start;

}

struct node \*addatend(struct node \*start, int data)

{

struct node \*p,\*tmp;

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

tmp->info=data;

p=start;

while(p->link!=NULL)

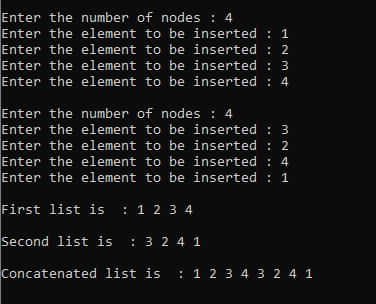
p=p->link;

p->link=tmp;

tmp->link=NULL;

return start;

}



**Task2**

/\* Program to insert in a sorted list \*/

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

struct Node {

int data;

struct Node\* next;

};

void sortedInsert(struct Node\*\* head\_ref,

struct Node\* new\_node)

{

struct Node\* current;

/\* Special case for the head end \*/

if (\*head\_ref == NULL

|| (\*head\_ref)->data

>= new\_node->data) {

new\_node->next = \*head\_ref;

\*head\_ref = new\_node;

}

else {

/\* Locate the node before

the point of insertion \*/

current = \*head\_ref;

while (current->next != NULL

&& current->next->data < new\_node->data) {

current = current->next;

}

new\_node->next = current->next;

current->next = new\_node;

}

}

/\* A utility function to create a new node \*/

struct Node\* newNode(int new\_data)

{

/\* allocate node \*/

struct Node\* new\_node

= (struct Node\*)malloc(

sizeof(struct Node));

/\* put in the data \*/

new\_node->data = new\_data;

new\_node->next = NULL;

return new\_node;

}

/\* Function to print linked list \*/

void printList(struct Node\* head)

{

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

}

/\* Driver program to test count function\*/

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

struct Node\* new\_node = newNode(20);

sortedInsert(&head, new\_node);

new\_node = newNode(17);

sortedInsert(&head, new\_node);

new\_node = newNode(3);

sortedInsert(&head, new\_node);

new\_node = newNode(18);

sortedInsert(&head, new\_node);

new\_node = newNode(27);

sortedInsert(&head, new\_node);

printf("Created Linked List\n ");

printList(head);

return 0;

}



**Task3**

#include<stdio.h>

#include<stdlib.h>

/\* Link list node \*/

struct Node

{

int data;

struct Node\* next;

};

/\* Function to get the middle of the linked list\*/

void printMiddle(struct Node \*head)

{

struct Node \*slow\_ptr = head;

struct Node \*fast\_ptr = head;

if (head!=NULL)

{

while (fast\_ptr != NULL && fast\_ptr->next != NULL)

{

fast\_ptr = fast\_ptr->next->next;

slow\_ptr = slow\_ptr->next;

}

printf("The middle element is [%d]\n\n", slow\_ptr->data);

}

}

void push(struct Node\*\* head\_ref, int new\_data)

{

/\* allocate node \*/

struct Node\* new\_node =

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

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to point to the new node \*/

(\*head\_ref) = new\_node;

}

// A utility function to print a given linked list

void printList(struct Node \*ptr)

{

while (ptr != NULL)

{

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

ptr = ptr->next;

}

printf("NULL\n");

}

/\* Driver program to test above function\*/

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

int i,n,val;

printf("How many elements do u want to enter?");

scanf("%d",&n);

if(n==0)

printf("Median of empty list is [0]");

for (i=n; i>0; i--)

{

printf("Enter value;");

scanf("%d",&val);

push(&head, val);

if(i==1)

{

printList(head);

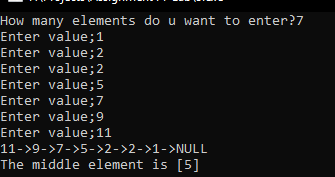
printMiddle(head);

}

}

return 0;

}



**Task4**

/\*\*

\* C program to reverse a Singly Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

/\* Structure of a node \*/

struct node {

int data; //Data part

struct node \*next; //Address part

}\*head;

/\* Functions used in the program \*/

void createList(int n);

void reverseList();

void displayList();

int main()

{

int n, choice;

/\*

\* Create a singly linked list of n nodes

\*/

printf("Enter the total number of nodes: ");

scanf("%d", &n);

createList(n);

printf("\nData in the list \n");

displayList();

/\*

\* Reverse the list

\*/

printf("\nPress 1 to reverse the order of singly linked list\n");

scanf("%d", &choice);

if(choice == 1)

{

reverseList();

}

printf("\nData in the list\n");

displayList();

return 0;

}

/\*

\* Create a list of n nodes

\*/

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

if(n <= 0)

{

printf("List size must be greater than zero.\n");

return;

}

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

/\*

\* If unable to allocate memory for head node

\*/

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

/\*

\* Read data of node from the user

\*/

printf("Enter the data of node 1: ");

scanf("%d", &data);

head->data = data; // Link the data field with data

head->next = NULL; // Link the address field to NULL

temp = head;

/\*

\* Create n nodes and adds to linked list

\*/

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

{

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

/\* If memory is not allocated for newNode \*/

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link the data field of newNode with data

newNode->next = NULL; // Link the address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

/\*

\* Reverse the order of nodes of a singly linked list

\*/

void reverseList()

{

struct node \*prevNode, \*curNode;

if(head != NULL)

{

prevNode = head;

curNode = head->next;

head = head->next;

prevNode->next = NULL; // Make first node as last node

while(head != NULL)

{

head = head->next;

curNode->next = prevNode;

prevNode = curNode;

curNode = head;

}

head = prevNode; // Make last node as head

printf("SUCCESSFULLY REVERSED LIST\n");

}

}

/\*

\* Display entire list

\*/

void displayList()

{

struct node \*temp;

/\*

\* If the list is empty i.e. head = NULL

\*/

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

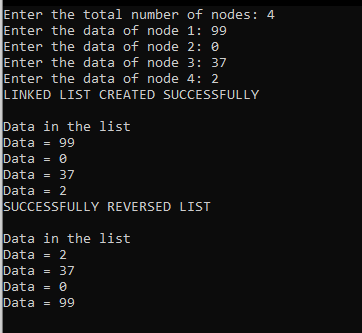
printf("Data = %d\n", temp->data); // Print the data of current node

temp = temp->next; // Move to next node

}

}

}



**Task5i**

#include <stdio.h>

#include <stdlib.h>

/\*

\* Basic structure of Node

\*/

struct node {

int data;

struct node \* prev;

struct node \* next;

}\*head, \*last;

/\*

\* Function used in this program

\*/

void createList(int n);

void displayListFromFirst();

void displayListFromEnd();

int main()

{

int n, choice;

head = NULL;

last = NULL;

printf("Enter the number of nodes you want to create: ");

scanf("%d", &n);

createList(n); // Create list of n nodes

displayListFromFirst();

return 0;

}

/\*\*

\* Create a doubly linked list of n nodes.

\* @n Number of nodes to be created

\*/

void createList(int n)

{

int i, data;

struct node \*newNode;

if(n >= 1)

{

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

if(head != NULL)

{

printf("Enter data of 1 node: ");

scanf("%d", &data);

head->data = data;

head->prev = NULL;

head->next = NULL;

last = head;

/\*

\* Create rest of the n-1 nodes

\*/

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

{

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

if(newNode != NULL)

{

printf("Enter data of %d node: ", i);

scanf("%d", &data);

newNode->data = data;

newNode->prev = last; // Link new node with the previous node

newNode->next = NULL;

last->next = newNode; // Link previous node with the new node

last = newNode; // Make new node as last/previous node

}

else

{

printf("Unable to allocate memory.");

break;

}

}

printf("\nDOUBLY LINKED LIST CREATED SUCCESSFULLY\n");

}

else

{

printf("Unable to allocate memory");

}

}

}

/\*\*

\* Displays the content of the list from beginning to end

\*/

void displayListFromFirst()

{

struct node \* temp;

int n = 1;

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

printf("\n\nDATA IN THE LIST:\n");

while(temp != NULL)

{

printf("DATA of %d node = %d\n", n, temp->data);

n++;

/\* Move the current pointer to next node \*/

temp = temp->next;

}

}

}

/\*\*

\* Display the content of the list from last to first

\*/

void displayListFromEnd()

{

struct node \* temp;

int n = 0;

if(last == NULL)

{

printf("List is empty.");

}

else

{

temp = last;

printf("\n\nDATA IN THE LIST:\n");

while(temp != NULL)

{

printf("DATA of last-%d node = %d\n", n, temp->data);

n++;

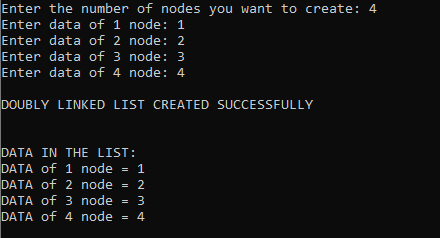
/\* Move the current pointer to previous node \*/

temp = temp->prev;

}

}

}



**Task5ii**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct node {

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

int getMaxNode(struct node \*stnode);

void displayDlList();

int main()

{

int n;

stnode = NULL;

ennode = NULL;

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

scanf("%d", &n);

DlListcreation(n);

displayDlList();

printf("\n The Minimum Value in the Linked List : %d\n\n", getminNode(stnode));

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : "); // assigning data in the first node

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

// putting data for rest of the nodes

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

{

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

if(fnNode != NULL)

{

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

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode; // new node is linking with the previous node

fnNode->nextptr = NULL;

ennode->nextptr = fnNode; // previous node is linking with the new node

ennode = fnNode; // assign new node as last node

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

int getminNode(struct node \*stnode)

{

if(stnode == NULL){

printf(" User message : Invalid Input !!!!\n");

return INT\_MIN;

}

int min = stnode->num;

while(stnode != NULL){

if(stnode->num < min){

min = stnode->num;

}

stnode = stnode->nextptr;

}

return min;

}

void displayDlList()

{

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");

while(tmp != NULL)

{

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

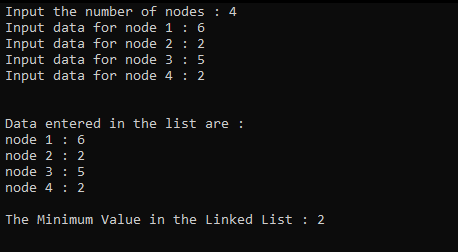
n++;

tmp = tmp->nextptr; // current pointer moves to the next node

}

}

}



**Task5iii**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct node {

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

int getMaxNode(struct node \*stnode);

void displayDlList();

void displayDlList1(int a);

int main()

{

int n;

stnode = NULL;

ennode = NULL;

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

scanf("%d", &n);

DlListcreation(n);

displayDlList();

printf("\n The Minimum Value in the Linked List : %d\n\n", getminNode(stnode));

printf("Min Node deleted successfully ; New LinkedList;\n");

displayDlList1(getminNode(stnode));

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : "); // assigning data in the first node

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

// putting data for rest of the nodes

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

{

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

if(fnNode != NULL)

{

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

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode; // new node is linking with the previous node

fnNode->nextptr = NULL;

ennode->nextptr = fnNode; // previous node is linking with the new node

ennode = fnNode; // assign new node as last node

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

int getminNode(struct node \*stnode)

{

if(stnode == NULL){

printf(" User message : Invalid Input !!!!\n");

return INT\_MIN;

}

int min = stnode->num;

while(stnode != NULL){

if(stnode->num < min){

min = stnode->num;

}

stnode = stnode->nextptr;

}

return min;

}

void displayDlList()

{

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");

while(tmp != NULL)

{

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

n++;

tmp = tmp->nextptr; // current pointer moves to the next node

}

}

}

void displayDlList1(int a)

{

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");

while(tmp != NULL)

{

if(tmp->num==a)

{

}

else

{

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

}

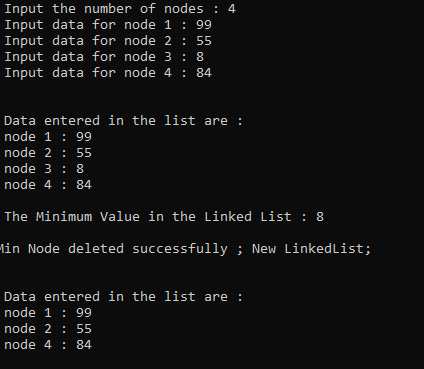
n++;

tmp = tmp->nextptr; // current pointer moves to the next node

}

}

}



**Task5iv**

/\*\*

\* C program to reverse a Doubly linked list

\*/

#include <stdio.h>

#include <stdlib.h>

/\*

\* Basic structure of Node

\*/

struct node {

int data;

struct node \* prev;

struct node \* next;

}\*head, \*last;

/\*

\* Functions used in this program

\*/

void createList(int n);

void displayList();

void reverseList();

int main()

{

int n, data, choice=1;

head = NULL;

last = NULL;

printf("Enter the total number of nodes in list: ");

scanf("%d", &n);

createList(n);

reverseList();

displayList();

return 0;

}

/\*\*

\* Creates a doubly linked list of n nodes.

\* @n Number of nodes to be created

\*/

void createList(int n)

{

int i, data;

struct node \*newNode;

if(n >= 1)

{

/\*

\* Create and link head node

\*/

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

printf("Enter data of 1 node: ");

scanf("%d", &data);

head->data = data;

head->prev = NULL;

head->next = NULL;

last = head;

/\*

\* Create and link rest of the n-1 nodes

\*/

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

{

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

printf("Enter data of %d node: ", i);

scanf("%d", &data);

newNode->data = data;

newNode->prev = last; // Link new node with the previous node

newNode->next = NULL;

last->next = newNode; // Link previous node with the new node

last = newNode; // Make new node as last/previous node

}

printf("\nDOUBLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

/\*\*

\* Display the content of the list from beginning to end

\*/

void displayList()

{

struct node \* temp;

int n = 1;

if(head == NULL)

{

printf("List is empty.\n");

}

else

{

temp = head;

printf("DATA IN THE LIST:\n");

while(temp != NULL)

{

printf("DATA of %d node = %d\n", n, temp->data);

n++;

/\* Move pointer to next node \*/

temp = temp->next;

}

}

}

/\*\*

\* Reverse order of the doubly linked list

\*/

void reverseList()

{

struct node \*current, \*temp;

current = head;

while(current != NULL)

{

/\*

\* Swap the previous and next address fields of current node

\*/

temp = current->next;

current->next = current->prev;

current->prev = temp;

/\* Move the current pointer to next node which is stored in temp \*/

current = temp;

}

/\*

\* Swap the head and last pointers

\*/

temp = head;

head = last;

last = temp;

printf("LIST REVERSED SUCCESSFULLY.\n");

}

