(EXERCISE 1- EXERCISE 9)

#include<bits/stdc++.h>

using namespace std;

#include <stdlib.h>

#include <conio.h>

struct node

{

int data;

struct node \*next;

struct node \* prev;

}\*head, \*last;;

void createNodeList(int n);

void displayList();

void createList(int n)

{

int i, data;

struct node \*newNode;

if(n>=1)

{

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

cout<<"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));

cout<<"Enter data of "<<i<< " node: ";

cin>>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

}

cout<<"\n LINKED LIST CREATED SUCCESSFULLY"<<endl;

}

}

void insertAtBeginning(int data)

{

struct node \* newNode;

if(head == NULL)

{

cout<<"Error, List is Empty!"<<endl;

}

else

{

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

newNode->data = data;

newNode->next = head; // Point to next node which is currently head

newNode->prev = NULL; // Previous node of first node is NULL

/\* Link previous address field of head with new node \*/

head->prev = newNode;

/\* Make the new node as head node \*/

head = newNode;

cout<<"NODE INSERTED SUCCESSFULLY AT THE BEGINNING OF THE LIST"<<endl;

}

}

void insertAtEnd(int data)

{

struct node \* newNode;

if(last == NULL)

{

cout<<"Error, List is empty!"<<endl;

}

else

{

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

newNode->data = data;

newNode->next = NULL;

newNode->prev = last;

last->next = newNode;

last = newNode;

cout<<"NODE INSERTED SUCCESSFULLY AT THE END OF LIST"<<endl;

}

}

void deleteBeginning()

{

if(head == NULL)

cout<<"List is Empty!!! Deletion not possible!!!";

else

{

struct node \*temp = head;

if(temp -> next == head)

{

head = NULL;

free(temp);

}

else

{

head = head -> next;

free(temp);

}

cout<<"\nDeletion success!! (•\_•)";

}

}

void deleteEnd()

{

struct node \*temp1,\*temp2;

if(head == NULL)

cout<<"List is Empty!!! Deletion not possible!! (•\_•)";

else

{

temp1= head;

temp2=head;

while(temp1 -> next != NULL)

{

temp2=temp1;

temp1=temp1->next;

}

if(temp1==head)

{

head=NULL;

}

else

temp2->next=NULL;

}

free(temp1);

cout<<"\nDeletion success!!!";

}

void deleteNodeFromAny(int delValue)

{

if(head == NULL)

cout<<"List is Empty!! Deletion not possible. (•\_•)";

else

{

struct node \*temp1 = head, \*temp2;

while(temp1-> data != delValue)

{

if(temp1-> next == head)

{

cout<<"\nGiven node is not found in the list!!!";

}

else

{

temp2 = temp1;

temp1 = temp1->next;

}

}

if(temp1 -> next == head)

{

head = NULL;

free(temp1);

}

else

{

if(temp1 == head)

{

temp2 = head;

while(temp2 -> next != head)

temp2 = temp2 -> next;

head = head -> next;

temp2 -> next = head;

free(temp1);

}

else

{

if(temp1->next==head)

{

temp2->next=head;

}

else

{

temp2->next=temp1->next;

}

free(temp1);

}

}

cout<<"\nDeletion success!!!";

}

}

void reverseList()

{

struct node \*prevNode, \*curNode;

if(head != NULL)

{

prevNode = head;

curNode = head->next;

head = head->next;

prevNode->next = NULL;

while(head != NULL)

{

head = head->next;

curNode->next = prevNode;

prevNode = curNode;

curNode = head;

}

head = prevNode;

cout<<"SUCCESSFULLY REVERSED LIST"<<endl;

}

}

void removeDublicates()

{

struct node \*p, \*q, \*prev, \*temp;

p = q = prev = head;

q = q->next;

while (p != NULL)

{

while (q != NULL && q->data != p->data)

{

prev = q;

q = q->next;

}

if (q == NULL)

{

p = p->next;

if (p != NULL)

{

q = p->next;

}

}

else if (q->data == p->data)

{

prev->next = q->next;

temp = q;

q = q->next;

free(temp);

}

}

cout<<"Duplicate Number delete done \n"<<endl;

}

void displayList()

{

struct node \* temp;

int n = 1;

if(head == NULL)

{

cout<<"List is empty."<<endl;

}

else

{

temp = head;

cout<<"DATA IN THE LIST:"<<endl;

while(temp != NULL)

{

cout<<"DATA of "<<n <<" node = "<<temp->data<<endl;

n++;

temp = temp->next; /\* Move the current pointer to next node \*/

}

}

}

int main()

{

int n,y;

struct node \*ND, \*tmp;

int data, location, i,j,p;

start:

{

cout<<"\n\t(0) Create List \n\t(1) Insert new node \n\t(2) Insert node at beginning \n\t(3) Insert at any position at NODELIST \n\t(4) Delete Node from last position \n\t(5) Delete Node from beginning \n\t(6) Delete Node from any position \n\t(7) Reverse Linked list \n\t(8) Remove Duplicates \n\t(9) Display list \n\t(10) Exit";

cout<<"\n Enter Your choice : ";

cin>>y;

}

switch(y)

{

case 0:

system("cls");

{

cout<<"Enter the total number of nodes in list: ";

cin>>n;

createList(n);

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 1:

system("cls");

{

cout<<" Input the number of nodes : ";

cin>>n;

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

if(head == NULL)

{

cout<<" Memory can not be allocated.";

}

else

{

cout<<" Input data for node 1 : ";

cin>>data;

head->data = data;

head->next = NULL;

tmp = head;

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

{

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

if(ND == NULL)

{

cout<<" Memory can not be allocated.";

break;

}

else

{

cout<<" Input data for node "<<i<< " : ";

cin>>data;

ND->data = data;

ND->next = NULL;

tmp->next = ND;

tmp = tmp->next;

}

}

}

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 2:

system("cls");

{

cout<<"Enter data of first node : ";

cin>>data;

insertAtBeginning(data);

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 3:

system("cls");

{

cout<<"Enter data of last node : ";

cin>>data;

insertAtEnd(data);

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 4:

system("cls");

{

deleteEnd();

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 5:

system("cls");

{

deleteBeginning();

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 6:

system("cls");

{

cout<<"Enter the location : ";

cin>>location;

deleteNodeFromAny(location);

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 7:

system("cls");

{

reverseList();

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 8:

system("cls");

{

removeDublicates();

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 9:

system("cls");

{

displayList();

}

cout<<"\n\t..::Enter any key to go Menu::.."<<endl;

getch();

system("cls");

goto start;

case 10:

{

break;

}

}

}