**Name : Aditi Sant , AI&DS-B1 batch , Roll no.: 23211**

**EXPERIMENT NO:7**

#include <iostream>

#include <string.h>

using namespace std;

//Node

struct node {

int prn;

string name;

struct node \*next;

};

//Linked list

class list {

node \*head, \*temp;

public:

list() {

head = NULL;

}

node \*create(int val, string n);

void insertEnd();

void insertBeg();

void deleteAt(int i);

void insertAt(int i);

void display();

int count();

void concatenate(list A,list B);

void op();

};

//Create

node\* list::create(int val, string n)

{

temp = new(struct node);

if (temp == NULL)

{

cout<<"Memory Allocation Failed!"<<endl;

return 0;

}

else

{

temp -> prn = val;

temp -> name = n;

temp -> next = NULL;

return temp;

}

}

//Insert End

void list::insertEnd() {

int val;

string n;

cout<<"Enter PRN: ";

cin>>val;

cout<<"Enter Name: ";

cin>>n;

struct node \*t = head;

temp = create(val,n);

if (head == NULL)

{

head = temp;

head -> next = NULL;

}

else

{

while ((t -> next) != NULL)

{

t = t -> next;

}

temp -> next = NULL;

t -> next = temp;

cout<<"Element Inserted at Last"<<endl;

}

}

//Insert At

void list::insertAt(int i)

{

int val,pos = i - 1,counter = 1;

string n;

struct node \*ptr;

struct node \*t = head;

while ((t -> next) != NULL)

{

//loop to count number of items in linked list.

t = t -> next;

counter++;

}

t = head;

//traverse pointer is pointed to head again.

if (i == 1)

{

//equivalent to insert at start.

insertBeg();

}

else if (pos > counter || i <= 0)

{

//if position is greater than the actual linked list.

cout<<"Entered position is out of scope."<<endl;

}

else

{

//insert at required position.

cout<<"Enter PRN: ";cin>>val;

cout<<"Enter Name: ";

cin>>n;

temp = create(val,n);

while (pos--)

{

ptr = t;

t = t -> next;

}

temp -> next = t;

ptr -> next = temp;

cout<<"Member Inserted at Position: "<<i<<endl;

}

}

//Delete At

void list::deleteAt(int i)

{

int val,pos = i - 1,counter = 1;

string n;

struct node \*ptrl,\*ptrr;

struct node \*t = head;

while ((t -> next) != NULL)

{

t = t -> next;

counter++;

}

t = head;

if (i == 1)

{

ptrl = head;

head = head -> next;

delete ptrl;

}

else if (pos > counter || i <= 0)

{

cout<<"Entered member doesn't exist."<<endl;

}

else

{

while (pos--)

{

ptrl = t;

t = t -> next;

ptrr = t -> next;

}

ptrl -> next = ptrr;

delete t;

cout<<"Member Deleted at Position: "<<i<<endl;

}

}

//Insert Beg

void list::insertBeg()

{

int val;

string n;

cout<<"Enter PRN: ";

cin>>val;

cout<<"Enter Name: ";

cin>>n;

//v = val;

struct node \*t = head;

temp = create(val,n);

if (head == NULL)

{

head = temp;

head -> next = NULL;

}

else

{

temp -> next = head;

head = temp;

cout<<"We have a New President."<<endl;

}

}

//Display

void list::display() {

temp = head;

cout<<"President: ";

cout<< temp -> prn<<" — "<<temp -> name<<" -> ";

if(temp -> next != NULL)

{

temp = temp -> next;

}

while (temp -> next != NULL)

{

cout<< temp -> prn<<" — "<<temp -> name<<" -> ";

temp = temp -> next;

}

cout<<"Secretary: ";

cout<< temp -> prn<<" — "<<temp -> name<<" -> ";

cout<<"NULL"<<endl;

}

//Count

int list::count()

{

temp = head;

int ct = 0;

while (temp != NULL)

{

ct++;

temp = temp -> next;

}

return ct;

}

//Concatenate

void list::concatenate(list A,list B)

{

struct node \* last,\*last1;

node\* t = A.head;

while (t != NULL)

{

int val = t -> prn;

string n = t -> name;

temp = create(val,n);

if (head == NULL)

{

head = temp;head -> next = NULL;

last=head;

}

else

{

//temp -> next = NULL;

last -> next = t;

last=t;

}

t = t -> next;

}

last -> next = B.head;

t = B.head;

while (t != NULL)

{

int val = t -> prn;

string n = t -> name;

temp = create(val,n);

last -> next = temp;

last= temp;

t = t -> next;

}

last->next=NULL;

}

//Accept

void list::op()

{

while(1)

{

int choice;

cout<<"\nEnter: \n1. Add \n2. Delete \n3. Member's Count \n4. Display\n0. Prev Menu"<<endl;

cin>>choice;

switch(choice)

{

case 1://Add

{

char c;

cout<<"\nEnter: \nA. Add President \nB. Add Secretary \nC.Add Member"<<endl;

cin>>c;

switch(c) {

case 'A':

case 'a':

{

insertBeg();break;

}

case 'B':

case 'b':

{

insertEnd();break;

}

case 'C':

case 'c':

{

insertAt(2);break;

}

}

break;

}

case 2://Delete

{

char c;

cout<<"\nEnter: \nA. Delete President \nB. Delete Secretary\nC. Delete Member"<<endl;

cin>>c;

switch(c)

{

case 'A':

{

deleteAt(1);

cout<<"Club must have a President. Enter Details"<<endl;

insertBeg();

break;

}

case 'B':

{

deleteAt(count());

cout<<"Club must have a Secretary. Enter Details"<<endl;

insertEnd();

break;

}

case 'C':

{

int j;

cout<<"Enter Position for Deletion"<<endl;

cin>>j;

deleteAt(j);

break;

}

}

break;

}

case 3://Count

{

cout<<"Count: "<<count()<<endl;

break;

}

case 4://Display

{

if (head == NULL)

{

cout<<"NULL"<<endl;

break;

}

else

{

display();

break;

}

}

case 0://Prev Menu

{

return ;

}

}

}

}

//Main

int main() {

cout<<"Name : Aditi Sant , AI&DS-B1 batch , Roll no.: 23211"<<endl;

list L,X,Y;

int c;

while(1)

{

cout<<"Enter: \n1. List A \n2. List B \n3. Concatenate\n0. Exit"<<endl;

cin>>c;

switch(c)

{

case 1: cout<<"\nList A:"; X.op(); break;

case 2: cout<<"\nList B:"; Y.op(); break;

case 3: L.concatenate(X,Y); L.display(); break;

case 0: return 0;

}

}

}

**OUTPUT**



