**DSA LAB 4**

**PART B:**

#include<iostream>

#include<string>

using namespace std;

class Node {

public:

int ID;

string designation;

float salary;

int noofdays;

string company;

string name;

Node\* next ;

Node(int id, string desi, string nam, string com, float sal, int no) :ID(id), designation(desi), salary(sal), noofdays(no), company(com), name(nam),next(nullptr) {}

};

//linked list class

class Linkedlist {

Node\* head = nullptr;

Node\* newnode=nullptr;

Node\* current = nullptr;

Node\* prev = nullptr;

public:

void add(int roll, float sal, string nam, int noof, string com,string desi,int num)

{

current = head;

//at first

if (noof < 10)

{

cout << "\nNo of working days is less then 10. Data cannot be entered!";

return;

}

while (current != nullptr)

{

if (current->ID == roll)

{

cout << "\nEmployee ID already exist. Cannot enter record!.";

return;

}

}

newnode = new Node(roll, desi, nam, com,sal,noof);

if (num == 1)

{

if (head == nullptr)

{

head = newnode;

}

else

{

current = head;

head = newnode;

head->next = current;

}

}

//at last

if (num == 2)

{

if (head == nullptr)

{

head = newnode;

}

else

{

current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = newnode;

}

}

//at specific position

if (num == 3)

{

cin.ignore();

cout << "\nenter index:";

cin >> num;

for (int i = 1; i < num - 1 && current != NULL; i++)

{

if (current->next == NULL)

{

cout << "out of bound index " << i;

return;

}

current = current->next;

}

newnode->next = current->next;

current->next = newnode;

}

}

void print()

{

current = head;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n"<<current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//deletion function

void del()

{

cout << "\nEnter ID of employee to be deleted: ";

int num;

cin >> num;

bool flag=false;

while (current != nullptr)

{

if (current->next->ID == num)

{

flag = true;

break;

}

}

if (!flag)

{

cout << "\nNo record found!";

return;

}

prev = current;

current->next = current->next->next;

delete prev;

prev = nullptr;

}

//searching function

void search()

{

cout << "\nEnter ID of employee to be searched: ";

int num;

cin >> num;

bool flag = false;

while (current != nullptr)

{

if (current->next->ID == num)

{

flag = true;

break;

}

}

if (!flag)

{

cout << "\nNo record found!";

return;

}

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

//even odd

void even()

{

current = head;

cout << "\tEven employee ID data\n";

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

if (current->ID % 2 == 0)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

current = current->next;

}

cout << "\n\tODD employee ID data\n";

current = head;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

if (!(current->ID % 2 == 0))

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

current = current->next;

}

}

//skipp first two node print

void skip()

{

current = head;

current = current->next->next;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//mean employee salary

void mean()

{

current = head;

float avg=0;

int count = 0;

current = current->next->next;

while (current != nullptr)

{

avg += current->salary;

count++;

current = current->next;

}

cout << "\nMean employee salary: " << avg / count;

}

//update record

void update()

{

string name, desi, com;

float sal;

int id, noof, num;

Linkedlist obj;

cout << "Enter Employee ID to update record of Employee: ";

cin >> id;

while (current != nullptr)

{

if (current->ID == id)

{

cout << "\n Enter Name: ";

getline(cin, name);

cout << "Enter company name: ";

getline(cin, com);

cout << "Enter Designation: ";

getline(cin, desi);

cout << "Enter ID: ";

cin.ignore();

cin >> id;

cout << "Enter no of working days: ";

cin >> noof;

cout << "Enter salary: ";

cin >> sal;

cout << "Enter position at which u want to store data: \n1 for first postion\n2 for last position\n3 for specific position\n";

cin >> num;

obj.add(id, sal, name, noof, com, desi, num);

cin.ignore();

}

}

}

//sort

void sort()

{

current = head;

newnode = head;

while(newnode!=nullptr)

{

current = head;

while (current->next != nullptr)

{

if (current->salary > current->next->salary)

{

int data = current->ID;

current->ID = current->next->ID;

current->next->ID = data;

data = current->noofdays;

current->noofdays = current->next->noofdays;

current->next->noofdays = data;

float temp = current->salary;

current->salary = current->next->salary;

current->next->salary = temp;

string n = current->name;

current->name = current->next->name;

current->next->name = n;

n = current->designation;

current->designation = current->next->designation;

current->next->designation = n;

n = current->company;

current->company = current->next->company;

current->next->company = n;

}

current = current->next;

}

newnode = newnode->next;

}

}

//part b lab task

//calculating the mid and displaying linked list from mid to last and last to fist and first to mid.

void mid()

{

int count = 0;

current = head;

while (current != nullptr)

{

count++;

current = current->next;

}

count = count / 2;

int i = 0;

while (i <= count)

{

current = current->next;

i++;

}

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

i = 0; current = head;

while (i<=count)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//circular

void connect()

{

current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = head;

}

};

int main()

{

string name, desi, com;

float sal;

int id, noof, num;

Linkedlist obj;

int i = 1;

while (i<=4)

{

cout << "\n Enter Name: ";

getline(cin, name);

cout << "Enter company name: ";

getline(cin, com);

cout << "Enter Designation: ";

getline(cin, desi);

cout << "Enter ID: ";

cin.ignore();

cin >> id;

cout << "Enter no of working days: ";

cin >> noof;

cout << "Enter salary: ";

cin >> sal;

cout << "Enter position at which u want to store data: \n1 for first postion\n2 for last position\n3 for specific position\n";

cin >> num;

obj.add(id, sal, name, noof, com, desi, num);

cin.ignore();

i++;

}

//i have implemented all the functions above. program is not running due to some input issue acc to me. eacch function is implemented in above linkedlist class

system("pause");

}#include<iostream>

#include<string>

using namespace std;

class Node {

public:

int ID;

string designation;

float salary;

int noofdays;

string company;

string name;

Node\* next ;

Node(int id, string desi, string nam, string com, float sal, int no) :ID(id), designation(desi), salary(sal), noofdays(no), company(com), name(nam),next(nullptr) {}

};

//linked list class

class Linkedlist {

Node\* head = nullptr;

Node\* newnode=nullptr;

Node\* current = nullptr;

Node\* prev = nullptr;

public:

void add(int roll, float sal, string nam, int noof, string com,string desi,int num)

{

current = head;

//at first

if (noof < 10)

{

cout << "\nNo of working days is less then 10. Data cannot be entered!";

return;

}

while (current != nullptr)

{

if (current->ID == roll)

{

cout << "\nEmployee ID already exist. Cannot enter record!.";

return;

}

}

newnode = new Node(roll, desi, nam, com,sal,noof);

if (num == 1)

{

if (head == nullptr)

{

head = newnode;

}

else

{

current = head;

head = newnode;

head->next = current;

}

}

//at last

if (num == 2)

{

if (head == nullptr)

{

head = newnode;

}

else

{

current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = newnode;

}

}

//at specific position

if (num == 3)

{

cin.ignore();

cout << "\nenter index:";

cin >> num;

for (int i = 1; i < num - 1 && current != NULL; i++)

{

if (current->next == NULL)

{

cout << "out of bound index " << i;

return;

}

current = current->next;

}

newnode->next = current->next;

current->next = newnode;

}

}

void print()

{

current = head;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n"<<current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//deletion function

void del()

{

cout << "\nEnter ID of employee to be deleted: ";

int num;

cin >> num;

bool flag=false;

while (current != nullptr)

{

if (current->next->ID == num)

{

flag = true;

break;

}

}

if (!flag)

{

cout << "\nNo record found!";

return;

}

prev = current;

current->next = current->next->next;

delete prev;

prev = nullptr;

}

//searching function

void search()

{

cout << "\nEnter ID of employee to be searched: ";

int num;

cin >> num;

bool flag = false;

while (current != nullptr)

{

if (current->next->ID == num)

{

flag = true;

break;

}

}

if (!flag)

{

cout << "\nNo record found!";

return;

}

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

//even odd

void even()

{

current = head;

cout << "\tEven employee ID data\n";

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

if (current->ID % 2 == 0)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

current = current->next;

}

cout << "\n\tODD employee ID data\n";

current = head;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

if (!(current->ID % 2 == 0))

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

}

current = current->next;

}

}

//skipp first two node print

void skip()

{

current = head;

current = current->next->next;

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//mean employee salary

void mean()

{

current = head;

float avg=0;

int count = 0;

current = current->next->next;

while (current != nullptr)

{

avg += current->salary;

count++;

current = current->next;

}

cout << "\nMean employee salary: " << avg / count;

}

//update record

void update()

{

string name, desi, com;

float sal;

int id, noof, num;

Linkedlist obj;

cout << "Enter Employee ID to update record of Employee: ";

cin >> id;

while (current != nullptr)

{

if (current->ID == id)

{

cout << "\n Enter Name: ";

getline(cin, name);

cout << "Enter company name: ";

getline(cin, com);

cout << "Enter Designation: ";

getline(cin, desi);

cout << "Enter ID: ";

cin.ignore();

cin >> id;

cout << "Enter no of working days: ";

cin >> noof;

cout << "Enter salary: ";

cin >> sal;

cout << "Enter position at which u want to store data: \n1 for first postion\n2 for last position\n3 for specific position\n";

cin >> num;

obj.add(id, sal, name, noof, com, desi, num);

cin.ignore();

}

}

}

//sort

void sort()

{

current = head;

newnode = head;

while(newnode!=nullptr)

{

current = head;

while (current->next != nullptr)

{

if (current->salary > current->next->salary)

{

int data = current->ID;

current->ID = current->next->ID;

current->next->ID = data;

data = current->noofdays;

current->noofdays = current->next->noofdays;

current->next->noofdays = data;

float temp = current->salary;

current->salary = current->next->salary;

current->next->salary = temp;

string n = current->name;

current->name = current->next->name;

current->next->name = n;

n = current->designation;

current->designation = current->next->designation;

current->next->designation = n;

n = current->company;

current->company = current->next->company;

current->next->company = n;

}

current = current->next;

}

newnode = newnode->next;

}

}

//part b lab task

//calculating the mid and displaying linked list from mid to last and last to fist and first to mid.

void mid()

{

int count = 0;

current = head;

while (current != nullptr)

{

count++;

current = current->next;

}

count = count / 2;

int i = 0;

while (i <= count)

{

current = current->next;

i++;

}

cout << "ID\tNAME\tSALARY\tNO.OF.WORK.DAYS DESIGNATION\tCOMPANY\n";

while (current != nullptr)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

i = 0; current = head;

while (i<=count)

{

cout << "\n" << current->ID << "\t" << current->name << "\t" << current->salary << "\t" << current->noofdays << "\t" << current->designation << "\t" << current->company;

current = current->next;

}

}

//circular

void connect()

{

current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = head;

}

};

int main()

{

string name, desi, com;

float sal;

int id, noof, num;

Linkedlist obj;

int i = 1;

while (i<=4)

{

cout << "\n Enter Name: ";

getline(cin, name);

cout << "Enter company name: ";

getline(cin, com);

cout << "Enter Designation: ";

getline(cin, desi);

cout << "Enter ID: ";

cin.ignore();

cin >> id;

cout << "Enter no of working days: ";

cin >> noof;

cout << "Enter salary: ";

cin >> sal;

cout << "Enter position at which u want to store data: \n1 for first postion\n2 for last position\n3 for specific position\n";

cin >> num;

obj.add(id, sal, name, noof, com, desi, num);

cin.ignore();

i++;

}

//i have implemented all the functions above. program is not running due to some input issue acc to me. eacch function is implemented in above linkedlist class

system("pause");

}

**PART A:**

#include<iostream>

#include<string>

using namespace std;

class Node {

public:

int info;

Node\* next;

Node\* prev;

Node(int val):info(val),prev(nullptr),next(nullptr){}

};

class Linkedlist {

Node\* head = nullptr;

Node\* tail = nullptr;

Node\* current = nullptr;

Node\* newnode = nullptr;

public:

void add(int roll, int num)

{

current = head;

//at first

newnode = new Node(roll);

if (num == 1)

{

if (head == nullptr)

{

head = newnode;

tail = newnode;

tail->next = head;

}

else

{

newnode->next = head;

newnode -> prev = tail;

tail->next = newnode;

head->prev = newnode;

}

}

//at last

if (num == 2)

{

if (head == nullptr)

{

head = newnode;

tail = newnode;

tail->next = head;

}

else

{

tail->next = newnode;

newnode->prev = tail;

tail = newnode;

}

}

//at specific position

if (num == 3)

{

cin.ignore();

cout << "\nenter index:";

cin >> num;

for (int i = 1; i < num - 1 && current != NULL; i++)

{

if (current->next == NULL)

{

cout << "out of bound index " << i;

return;

}

current = current->next;

}

newnode->next = current->next;

newnode->prev = current;

current->next->prev = newnode;

current->next = newnode;

}

}

};