**Lab Description:**

In README.txt file

\*Not included on paper

**Main:**

#include "payroll.h" // PayRoll Class

#include "PayRollList.h" // Linked List class

#include <iostream> // cout, cin, endl

#include <fstream> // ifstream

/\*

\* Ryan Rosiak

\* COSC 220-001

\* Lab-3

\*/

int main() {

PayRollList list;

std::cout << "List 1: " << std::endl;

list.printPayChecks(); // Example of empty list print out

std::cout << "List 1 now being filled: " << std::endl; // First test of inserting into linked list and then outputting to console

list.insert("Ryan", 15.25, 10); // Basic insert of of the linked list

list.insert("John", 12, 25);

list.insert("Skylar", 13, 20);

list.insert("Bob", 20, 5);

PayRoll mary("Mary", 16, 22);

list.insert(mary);

list.printPayChecks();

PayRollList list2; // Second test of inserting into linked list and then outputting to console

std::cout << "List 2: Will show step by step of insert" << std::endl;

std::cout << "Insert 1: " << std::endl;

list2.insert("Jake", 45, 10);

list2.printPayChecks();

PayRoll martha("Martha", 5.45, 19);

PayRoll jenny("Jenny", 15.22, 15);

PayRoll james("James", 48, 12);

std::cout << "Insert 2: " << std::endl; // Step by step output of where the objects are getting inserted

list2.insert(martha);

list2.printPayChecks();

std::cout << "Insert 3: " << std::endl;

list2.insert(jenny);

list2.printPayChecks();

std::cout << "Insert 4: " << std::endl;

list2.insert(james);

list2.printPayChecks();

std::cout << "----------------File Portion----------------" << std::endl; // File is "employee.dat"

std::ifstream datafile;

datafile.open("employee.dat");

if (!datafile) {

std::cout << "Error opening file. \n";

return 1;

}

PayRoll parray[6];

std::string n;

double pr, hr;

for (int i = 0; i < 6; i++) { // Takes in input from structured .dat file

datafile >> n;

datafile >> pr;

datafile >> hr;

parray[i].setName(n);

parray[i].setRate(pr);

parray[i].setHours(hr);

}

datafile.close();

PayRollList list3; // Creates a third test list to store the array of objects taken from the file

for (int i = 0; i < 6; i++) {

list3.insert(parray[i]);

}

std::cout << "DataFile now apart of linked list: " << std::endl; // Shows linked list after array is input into it

list3.printPayChecks();

return 0;

}

**Sample Output:**

List 1:

The list is empty

List 1 now being filled:

Name: Bob

PayRate: 20

Hours: 5

Total Pay: 100

Name: Mary

PayRate: 16

Hours: 22

Total Pay: 352

Name: Ryan

PayRate: 15.25

Hours: 10

Total Pay: 152.5

Name: Skylar

PayRate: 13

Hours: 20

Total Pay: 260

Name: John

PayRate: 12

Hours: 25

Total Pay: 300

List 2: Will show step by step of insert

Insert 1:

Name: Jake

PayRate: 45

Hours: 10

Total Pay: 450

Insert 2:

Name: Jake

PayRate: 45

Hours: 10

Total Pay: 450

Name: Martha

PayRate: 5.45

Hours: 19

Total Pay: 103.55

Insert 3:

Name: Jake

PayRate: 45

Hours: 10

Total Pay: 450

Name: Jenny

PayRate: 15.22

Hours: 15

Total Pay: 228.3

Name: Martha

PayRate: 5.45

Hours: 19

Total Pay: 103.55

Insert 4:

Name: James

PayRate: 48

Hours: 12

Total Pay: 576

Name: Jake

PayRate: 45

Hours: 10

Total Pay: 450

Name: Jenny

PayRate: 15.22

Hours: 15

Total Pay: 228.3

Name: Martha

PayRate: 5.45

Hours: 19

Total Pay: 103.55

----------------File Portion----------------

DataFile now apart of linked list:

Name: Naomi

PayRate: 54.8

Hours: 40

Total Pay: 2192

Name: Nicholas

PayRate: 52.31

Hours: 12

Total Pay: 627.72

Name: Ryan

PayRate: 16.5

Hours: 25

Total Pay: 412.5

Name: Mary

PayRate: 12.9

Hours: 32

Total Pay: 412.8

Name: Jimmy

PayRate: 8.25

Hours: 20

Total Pay: 165

Name: Jarvis

PayRate: 5

Hours: 14

Total Pay: 70

**PayRoll.h:**

#ifndef PAYROLL\_H\_

#define PAYROLL\_H\_

#include <string>

class PayRoll {

private:

std::string name;

double payrate;

double hours;

public:

PayRoll(); // default ctor

PayRoll(std::string, double, double); // non-default ctor

double getRate(); // returns payrate

double getHours(); // returns hours

void setRate(double); // assigns payrate

void setName(std::string); // assigns name

void setHours(double); // assigns hours

double calculatePay(); // "getTotal()" returns the pay

void printInfo(); // prints info of all data members plus total pay

std::string getName(); // returns name

};

#endif

**PayRoll.cpp:**

#include "payroll.h" // Header file

#include <iostream> // cout, endl

#include <string>

/\* Ryan Rosiak

\*

\* Implementaion file for payroll.h:

\*

\*/

/\*

\* Default Constructor:

\* Initializes all members to default values.

\*/

PayRoll::PayRoll() {

name = "";

payrate = 0.0;

hours = 0.0;

}

/\*

\* Non-Default Constructor:

\* Takes a string, double, and another double as arguments and assigns them to

\* name, payrate, and hours.

\*/

PayRoll::PayRoll(std::string n, double pr, double hr) {

name = n;

payrate = pr;

hours = hr;

}

/\*

\* Payrate Setter:

\* Takes a double as an argument and assigns it to payrate

\*/

void PayRoll::setRate(double pr) {

payrate = pr;

}

/\*

\* Name Setter:

\* Takes a string as an argument and assigns it to name

\*/

void PayRoll::setName(std::string n) {

name = n;

}

/\*

\* Hours Setter:

\* Takes an double as an arugment and assigns it to hours

\*/

void PayRoll::setHours(double h) {

hours = h;

}

/\*

\* Calculate Function:

\* Calculates the total pay by returning the value of hours

\* multiplied by pay

\*/

double PayRoll::calculatePay() {

return hours \* payrate;

}

/\*

\* PrintInfo Function:

\* Displays all possible info in given class

\*/

void PayRoll::printInfo() {

std::cout << "Name: " << name << std::endl;

std::cout << "PayRate: " << payrate << std::endl;

std::cout << "Hours: " << hours << std::endl;

std::cout << "Total Pay: " << calculatePay() << std::endl;

}

/\*

\* Name Getter:

\* Returns a string that is the name of said object

\*/

std::string PayRoll::getName() {

return name;

}

/\*

\* PayRate Getter:

\* Returns the rate of the current object

\*/

double PayRoll::getRate() {

return payrate;

}

/\*

\* Hours Getter:

\* Returns the hours of the current object

\*/

double PayRoll::getHours() {

return hours;

}

**PayRollList.h:**

#ifndef \_PAYROLLLIST\_H

#define \_PAYROLLLIST\_H

#include "payroll.h"

#include <string>

#include <iostream>

class PayRollList {

private:

struct ListNode { // nodes for linked list

PayRoll p;

ListNode\* next;

};

ListNode\* head; // head of linked list

public:

PayRollList(); // Default ctor

~PayRollList(); // Destructor

void insert(std::string, double, double); // insert into linked list using 3 parameters

void insert(PayRoll); // insert into linked list using PayRoll object

void printPayChecks(); // Print function for all items in linked list

};

#endif

**PayRollList.cpp:**

#include "PayRollList.h"

/\*

\* Default Constructor

\* Takes no parameters and initializes head pointer to null

\*/

PayRollList::PayRollList() {

head = nullptr;

}

/\*

\* Insert Function:

\* Takes name, rate, and hours worked as parameters for a new ListNode.

\* Then calls overidden insert function to insert in order of payrate.

\*/

void PayRollList::insert(std::string n, double pr, double h) {

PayRoll input(n, pr, h);

insert(input);

}

/\*

\* Overidden insert:

\* Takes a PayRoll object directly.

\* Then, inserts the object into the linked list by order of payrate.

\*/

void PayRollList::insert(PayRoll newP) {

ListNode\* newNode = new ListNode;

newNode->p.setName(newP.getName());

newNode->p.setRate(newP.getRate());

newNode->p.setHours(newP.getHours());

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

if (head->p.getRate() < newNode->p.getRate()) {

ListNode\* temp = head;

head = newNode;

newNode = temp;

head->next = newNode;

return;

}

ListNode\* cursor = head;

while (cursor->next != nullptr) {

ListNode\* forward = cursor->next;

if ((cursor->p.getRate() > newNode->p.getRate()) && (forward->p.getRate() < newNode->p.getRate())) {

cursor->next = newNode;

newNode->next = forward;

return;

}

cursor = cursor->next;

}

cursor->next = newNode;

}

/\*

\* PrintPayChecks Function:

\* Prints each employee name and total pay out.

\*/

void PayRollList::printPayChecks() {

if (head == nullptr) {

std::cout << "The list is empty" << std::endl;

return;

}

ListNode\* cursor = head;

while (cursor) {

cursor->p.printInfo();

std::cout << std::endl;

cursor = cursor->next;

}

}

/\*

\* PayRollList Destructor:

\* Destroys dynamically allocated data

\*/

PayRollList::~PayRollList() {

if (head == nullptr) {

return;

}

ListNode\* cursor = head;

while (cursor) {

cursor = cursor->next;

delete head;

head = cursor;

}

}