CMPSC24 Midterm-I
Winter 2019

This exam is closed book, closed notes. You may use a one-page A4 size paper containing your notes. Write your name on your notes paper and all other sheets of the exam. No calculators or phones are allowed in the exam. **Write all your answers on the answer sheet. All exam sheets, notes paper and scratch paper must be turned in at the end of the exam. If you have a question, ask the instructor in writing using the scratch paper provided.**

By signing your name below, you are asserting that all work on this exam is yours alone, and that you will not provide any information to anyone else taking the exam. In addition, you are agreeing that you will not discuss any part of this exam with anyone who is not currently taking the exam in this room until after the exam has been returned to you. This includes posting any information about this exam on Piazza or any other social media. Discussing any aspect of this exam with anyone outside of this room constitutes a violation of the academic integrity agreement for CMPSC24.

Signature: _____

Name (please print clearly):_____

Umail address: _____@umail.ucsb.edu

Perm number: _____

You have 75 minutes to complete this exam. Work to maximize points. A hint for allocating your time: if a question is worth 10 points, spend no more than 5 minutes on it if a question is worth 20 points, spend no more than 10 minutes on it etc. If you don't know the answer to a problem, move on and come back later. Most importantly, stay calm. You can do this.

WRITE ALL YOUR ANSWERS IN THE PROVIDED ANSWER SHEET IN PEN OR DARK PENCIL!

DO NOT OPEN THIS EXAM UNTIL YOU ARE INSTRUCTED TO DO SO.
GOOD LUCK!

Name: _____

Q1 [20 points] The equation of a line has the form ***y = m*x + c,*** where ***m*** is the slope of the line and ***c*** is the y-intercept or more simply the intercept. ***m*** and ***c*** are constants for a given line, ***x*** and ***y*** are variables that represent the coordinates of any point on the line. ***x***, ***y,m*** and ***c*** all have real values. Below are plots that show examples of four different lines and their corresponding equations.

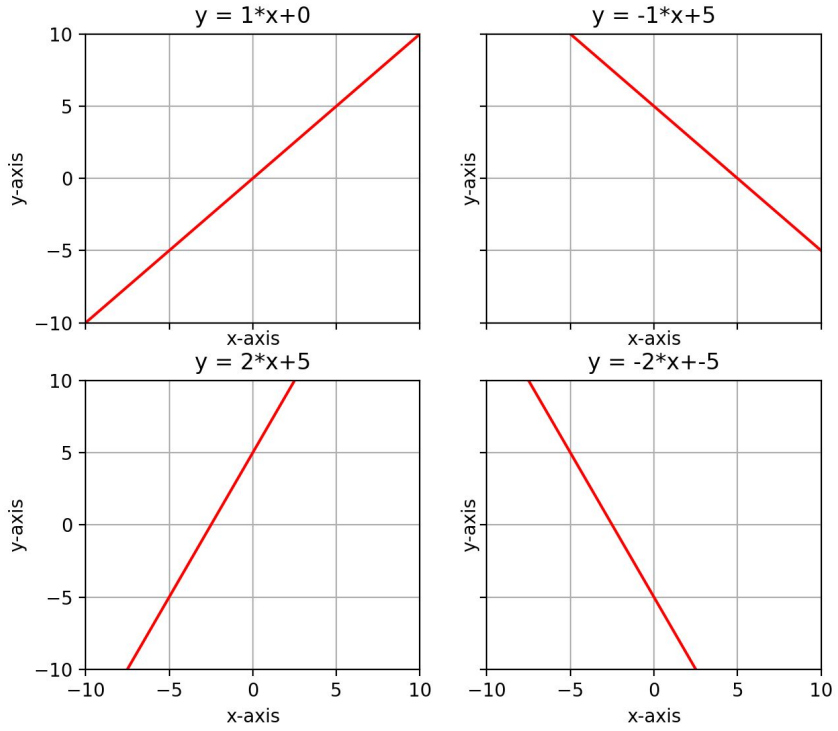

Figure 1: Examples of lines represented by the Line class

(i) (10 pts) Define and implement an Abstract Data Type called **`Line`** that represents the equation for a line. Your class implementation should satisfy the following requirements. To save space, consider implementing the member functions of the class inside the definition of the class.

a. (2 pts) Declare the private member variables of **`Line`** as **constants**.
b. (4 pts) Implement a parameterized constructor with default parameter values 0, 0 to initialize the slope and intercept of a new **`Line`** object. The first parameter should be the slope and the second parameter should be the intercept.
c. (4 pts) Implement the accessor functions **`getSlope`**( ) and **`getIntercept`**( ) to get the slope and intercept of a line. Make use of the keyword **`const`** as appropriate

For all the following questions assume that you have been given a correct implementation of the overloaded operator << that prints the equation of the line using the format ***y = m*x + c***. For example, if **`l1`** is an object of **`Line`** representing the line shown in the top right of Figure1, then **`cout<<l1;`** prints: ***y = -1.0*x + 5.0*** The prototype of the function is given below:

```
ostream& operator<<(ostream&out, const Line& li);
```

Name: _____

(ii) **(10 pts)** Implement the following operators as non-member functions:

a. Implement operator `||` that returns true if two lines are parallel otherwise returns false. Two lines are parallel if they have the same slope.
b. Implement the operator− that operates on two **`Line`** objects, and returns a new **`Line`** object whose slope (m) and intercept (c) are the difference of the slope and intercepts of two lines. For example, if **`l1`** is: ***`y = 2.0*x + 5.0`*** and **`l2`** is: ***`y = 8.0*x +10.0,`*** then **`l1 - l2`** should represent the line: ***`y = -6.0*x - 5.0`***

Q2 [10 points] Assume that you have a correct implementation of the class **`Line`** and the overloaded operators **`<<, ||, -`**

(i) **(8 pts)** For each of the following, state whether there will be any compile-time errors. If yes, explain why? If No, write the output of the code. You will be graded based on the description of the class **`Line`** and overloaded methods from Q1, and not your specific implementation. Please read Q1 carefully.

(a)(2 pts)

```
int main(){
        Line l1(2,7);
        cout<<l1.getSlope()<<", "<<l1.getIntercept()<<endl;
        return 0;
}
```

(b)(2 pts)

```
int main(){
        Line l1(2,7), l2(1,5), l3;
        l3 = l2;
        cout<<l3.getSlope()<<endl;
        return 0;
}
```

(c)(2 pts)

```
int main(){
        Line l1(2,7), l2(1,5);
        // Assume l1 is created at location 0x8000 and l2 at 0x9000
        Line& l3 = l1;
        Line* l4 = &l2;
        cout<<l3<<endl<<l4<<endl<<*l4;
        return 0;
}
```

Name: _____

```
(d)(2 pts)
 void foo(const Line& l1, const Line& l2){
    if(!(l1||l2)){
        cout<<l1 << l2 << "intersect"<< endl;
    } else{
        cout<<l1 << l2 <<" don't intersect"<<endl;
    }
 }

int main(){
    Line l1(2,7), l2(1,5), l3(2,10);
    foo(l1, l2);
    foo(l1, l3);
    return 0;
}
```

(ii) (2 pts) Consider the function given below that finds the point where two lines intersect:

```
void printIntersection(const& Line l1, const& Line l2){
      double x = -1*(l2-l1).getIntercept()/(l2-l1).getSlope();
      double y = l1.getSlope()* x + l1.getIntercept();
      cout<<"("<< x <<", "<< y <<")"<<endl;
}
```

Is the following implementation of the function equivalent in logic to the one above? Answer Yes or No and briefly justify your answer.

```
void printIntersection(const& Line l1, const& Line l2){
 double x =
  (l1.getIntercept()-l2.getIntercept())/(l2.getSlope()-l1.getSlope());
  double y = l2.getSlope()* x + l2.getIntercept();
  cout<<"("<< x <<", "<< y <<")"<<endl;
}
```

Name: _____

Q3 [20 points] Linked-list

Consider the definition of the class **`LinkedList`** used in the construction of a singly linked-list containing elements of type **`Player` (defined below).** The LinkedList class only stores nodes with **unique Player names**.

```
struct Player{
    string name;
    int score;
};

class LinkedList {
public:
     LinkedList(){head = tail = 0;}        //Default constructor
      ~LinkedList();                       // Destructor

       void append(const Player& p);
      LinkedList(const LinkedList& source); //Copy constructor
      LinkedList& operator=(const LinkedList& source); //Assignment
      bool find(const string& pname);

      // IMPLEMENT THIS PUBLIC FUNCTION:
      void incrementScore(const string& pname);

private:
      class Node{
        public:
           Player data;         // data element
           Node* next;         // pointer to the next node in the list
           Node (const Player& p){
                data = p;
                next = 0;
           }
      };
      Node* head; // pointer to the first node in the linked-list
      Node* tail; // pointer to the last node in the linked-list
      void clear(Node* h);
      //IMPLEMENT THESE PRIVATE FUNCTIONS
      bool findHelper(const string& pname, Node* h);
      void deleteMid(const string& pname);
};
```

Assume that you have been given correct implementations for the default constructor and the destructor

Name: _____

(i). (10 pts) For each of the implementations provided in parts a-e, selecting among the following options. SELECT ALL THAT APPLY

A. Implementation is correct
B. Implementation has a compile time error
C. Implementation has a run-time/logic error or memory leak

- If you select A, show the effect of calling the function on an example Linked List with a specific input. Use pointer diagrams to illustrate.
- If you select B, briefly describe the reason for the error
- If you select C, give an example of a specific input on a specific LinkedList where the function results in an incorrect output. Use pointer diagrams to illustrate your point!
- You are NOT expected to provide a correct implementation for these functions

(a) (2 pts) Append function

```
//Precondition: player p with name not currently in the linked list
//Post condition: Appends a new node at the end of the list
void LinkedList::append(const Player& p){
    Node* n = new Node(p);
    if(!head){
        head = n;
    } else{
        tail->next = n;
    }
    tail = n;
}
```

(b) (2 pts) Assume a correct implementation of the append function when evaluating the correctness of the code below:

```
LinkedList::LinkedList(const LinkedList& source){
      this->head = source.head;
      this->tail = source.tail;
      Node* p = source.head;
      while(p){
            this->append(p->data);
            p = p->next;
       }
 }
```

(c) (2 pts) Private clear function:

```
//Precondition: h is a pointer to a node in the linked list
//Postcondition: Recursively deletes all nodes starting at h
void LinkedList::clear(Node* h){
      if(!h) return;
      clear(h->next);
       delete h;
}
```

(d) (2 pts) Assume correct implementations of clear and the copy constructor when evaluating the code below:

```
LinkedList& LinkedList::operator=(const LinkedList& source){
     clear(this->head);
      LinkedList tmp(source);
     this->head = tmp->head;
     this-> tail = tmp->tail;
      return *this;
}
```

(e) (2 pts)

```
bool LinkedList::findHelper(const string& pname, Node* h){
      if(!h) return false;
      if(h->data == pname) return true;
       return findHelper(pname, h->next);
}
bool LinkedList::find(const string& pname){
      return findHelper(pname, head);
}
```

(ii) (5 pts) Implement the function **`deleteMid()`**. This function takes as input the name of a player and deletes the Node from the linked list with the given Player name.

```
//Precondition: The Node that needs to be deleted exists somewhere
between the first and last nodes and is not the first or the last node
in the linked list.
//Postcondition: Node with the given name is deleted. Since the linked
list only stores unique values, only one node is deleted
```

(iii) (5 pts) Implement the function **`incrementScore().`** This function takes as input the name of a player. If a player with the given name is already in the LinkedList, the function should increment the score of that player by one, otherwise it should append a new node at the end of the list with the given name and a score of 1. You may assume that a correct implementation of the append function is available.

THE END

Name: _____

Name: _____

Scratch Paper