University of Regina
Department of Computer Science
CS115 – Object-Oriented Design (Winter 2020)
Instructor: Gurmail Singh

Final Examination
Saturday, 19 December 2020 – 2:00pm - 5:00pm (3 hours)

By submitting answers, the student agrees to the following examination conditions and rules:

- All coding questions must be completed in **C++ language**.
- **Show all of your work** in each task and use clear, grammatically correct English for code comments. Write legible code with proper indentation and formatting!
- This is an **open book** examination. Use of external sources is allowed, but you cannot use ready outsourced/purchased solutions or any other person's assistance. Any suspected acts of academic misconduct on this exam will result in a grade of NR for the course until the investigating Dean completes their analysis and issues a verdict. A grade of NR will prevent you from registering in subsequent courses until the investigation is complete.
- You must maintain **confidentiality** of your work; do not provide any opportunity for others to see/copy any of your work. Failure to comply with this is cheating too.
- You have **10-minute grace period** for submission of your work in case of technical troubles. You are responsible for attending online examination on time and with full preparation. **Failure to submit work on time CANNOT be excused, no exceptions**!

Q1. [10 marks] You are given the following monthly report from SaskPower:

Client ID	Area	Bill Type	Power Units	Bill Amount	Effective cost per unit
1	(**R**)egina	(**I**)ndustrial	135	?	?
2	(**R**)egina	(**R**)esidential	62	?	?
3	(**S**)askatoon	(**R**)esidential	78	?	?
4	(**S**)askatoon	(**I**)ndustrial	100	?	?
5	(**M**)oose Jaw	(**I**)ndustrial	91	?	?
6	(**M**)oose Jaw	(**R**)esidential	53	?	?

Write a C++ program that will **complete the report** and **print it out** in console.
The **bill amount** has to be calculated using the following rules:

(**I**)ndustrial billing	(**R**)esidential billing

Base charge (same for both bill types):
For (**R**)egina: 20$
For (**S**)askatoon: 25$
For (**M**)oose Jaw: 15$	

For first 100 units: 0.3$ per unit After 100 units: 0.5$ per unit	For first 50 units: 0.15$ per unit After 50 units: 1$ per unit

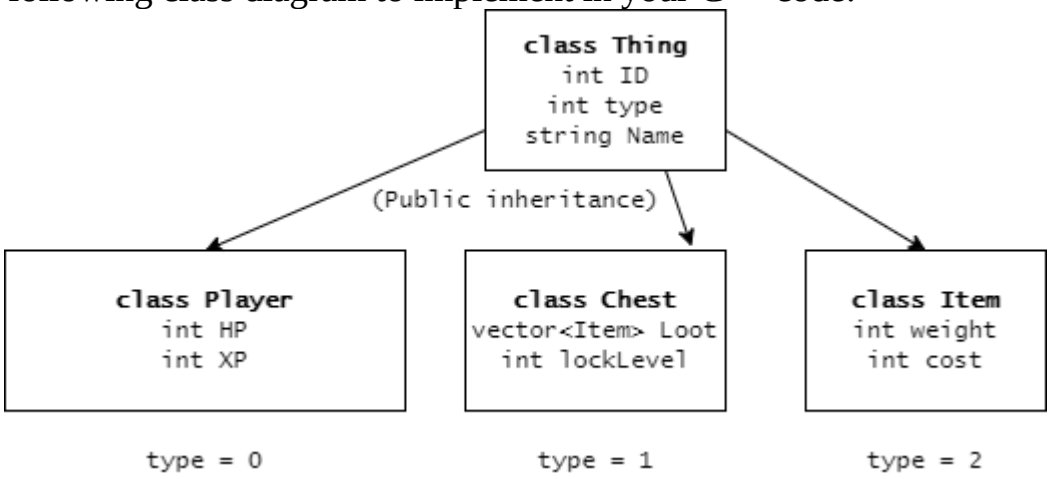
The **effective cost per unit** is calculated as **Bill Amount / Power Units**. After your program fills the table and prints it out, summarize the report with the following information:

- Show the Client ID with **the largest bill amount**
- Show the Client ID with **the smallest effective cost per unit**

Hints:

- Use the source code provided in Q1.cpp. Do NOT modify where commented.
  - Complete a billing function as other functions are already done for you

Q2. [10 marks] You have been hired to develop a single-player RPG game. Project manager gave you the following class diagram to implement in your C++ code:



Write a C++ program that will implement the above diagram of 4 classes, where **`Thing`** class is **publicly inherited** by the rest of them. To test the program functionality, you will have to do the following:

1. Create a **`Player`** class instance with the following characteristics:
   a. ID = 0
   b. Type = 0 (as shown on the diagram)
   c. Name = "Player 1"
   d. HP = 35
   e. XP = 0
2. Create an **`Item`** class instance with the following characteristics:
   a. ID = 1
   b. Type = 2 (as shown on the diagram)
   c. Name = "Iron club"
   d. Weight = 10
   e. Cost = 1
3. Create a **`Chest`** class instance with the following characteristics:
   a. ID = 2
   b. Type = 1 (as shown on the diagram)
   c. Name = "Wooden chest"
   d. Vector of **`Loot`** should contain the **Iron Club** that you created in the previous step.
   e. lockLevel = 5
4. Print the information about every created instance, based on what it is. Show all existing data fields per instance. (Note: all instances you created will be of **`Thing-like`** class by means of inheritance)

Hints:

- Use the source code provided in Q2.cpp. Do NOT modify where commented.
- No extra member functions (i.e. getters/setters) are required, all members may be public
  - You can print instance info as you go or afterwards via a function – at your choice

Q3. [10 marks] Write a ***struct House*** that has two member variables: string owner, int houseNumber. Create a dynamic linked list (in ascending order with respect to owner, that is, first name of owner) containing three instances of the ***struct House***:

- The first instance parameters would be: "Alice", 1
- The second instance parameters would be: "Bob", 2
- The third instance parameters would be: "Caren", 3

Write functions to insert/remove the nodes of the linked list, and a function to print all elements of a linked list. When you have all functions written and ready, do the following:

- Add a new instance: "Daryl", 4
- Remove all instances that have lowercase "e" in the first name field
- Print the updated linked list

Hints:

- Use the source code provided in Q3.cpp. Do NOT modify where commented.
- You CANNOT use the #include<list>

Q4. [10 marks] Consider the ***struct House*** as defined in Question 3 above. Declare a stack of instances of the struct. Implement the following in order:

(i) push following four elements to the stack
- "Paul", 19
- "John", 22
- "Patel", 18
- "Abdul", 29

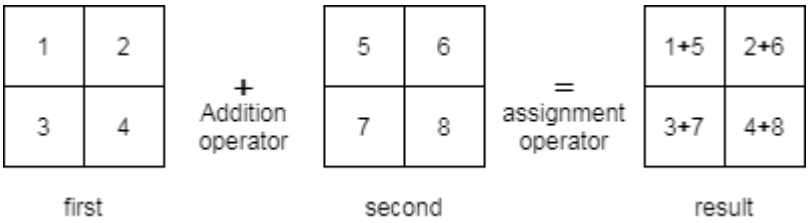
(ii) pop the last element from the stack
(iii) print the size of the stack
(iv) print the stack.

Hints:

- Use the source code provided in Q4.cpp. Do NOT modify where commented.
- No extra member functions (i.e. getters/setters) are required, all members may be public
- Don't prompt the user for the inputs, **hardcode required inputs** in the .cpp file

Q5. [10 marks] You are given a C++ class code describing a 2x2 matrix of integer values. Make functions for this class for **operator overloading** with the following operators:
+ (add), **–** (subtract). The **=** (assignment) is already done for you.

It should work like this:



All operations are to be done on respective elements of the two matrices (ex. for addition `result[i][j] = first[i][j] + second[i][j]` and so on for every **`i`** and **`j`**).
Test your program on the two matrices shown on the diagram above for these arithmetic operations that you have overloaded. Matrices `first` and `second` should remain unchanged.
The output should contain the following information:

- Result1 = first + second = ?
- Result2 = first – second = ?

Hints:

- Use the source code provided in Q5.cpp. Do NOT modify where commented.
- No extra member functions (i.e. getters/setters) are required, all members may be public

Examination Submission

Please try to submit ANY work done, even if it is not 100% finished!

Submit the answers to URCourses under "Final Exam" submission page.

You are expected to submit the following items:

- Completed source codes for Q1 through Q5. (five .cpp files)

You must use all available software to create a well-formatted, high quality submission. **If your submission cannot be read smoothly by the marker, your submission may receive reduced marks (possibly reduced to zero).**