CSCI 1300 Introduction to Computer Programming
Instructor: Fleming/ Gupta

Homework 8: Choose Project
Due Monday, March 19th by 6 pm

Homework 8: Class files & Code Skeleton
Due Sunday, April 8th by 6 pm

Homework 8: Final Deliverables
Due Sunday, April 22nd by 6 pm (no bonus available)

Homework 8: Interview Grading
begins on Monday, April 23rd

Homework 8: Project Report
Due Sunday, April 29th by 6 pm

This homework is a **project** and is worth **15%** of your overall grade.

Your completed project will be due **Sunday, April 22nd, 2018 by 6:00 pm**. The minimum requirements for the project can be found at the end of the project description.

OREGON TRAIL

For the Final Project you will implement a text based adventure game in C++. The game tries to follow closely the popular "**Oregon Trail**" game from the 1970s and 80s. Here are the original instructions:

```
THIS PROGRAM SIMULATES A TRIP OVER THE OREGON TRAIL FROM INDEPENDENCE
MISSOURI TO OREGON CITY, OREGON IN 1847. YOUR FAMILY OF FIVE WILL
COVER THE 2040 MILE OREGON TRAIL IN 5-6 MONTHS --- IF YOU MAKE IT
ALIVE."
```

The goal is to travel from Independence, Missouri to Oregon City (2040 miles) by the end of fall (November 30th, 1847). However, the trail is arduous. Each day costs you food and health. You can hunt and rest, but you have to get there before winter!

Suggestion: as you start reading through the project description, follow along the video of one of the later versions of the Oregon Trail game from 1985. It will help you understand which tasks you need to accomplish and in which order. https://youtu.be/FfbGEP087HM
You can also play the game in your browser: https://classicreload.com/oregon-trail.html

The game can be summarized by six core concepts:

1. The player buys supplies before starting the journey to Oregon.
2. There are opportunities to hunt for food along the way.
3. There are opportunities to make purchases at forts along the route.
4. The player must manage the level of supplies to avoid running out.
5. Misfortunes frequently occur.
6. The game ends when the player (and some of the companions) reach Oregon, when the player (and all the companions) die along the way, or when the player runs out of vital supplies (oxen, food, wagon parts).

Initial conditions:

- At the beginning of the game, the player (the leader of the expedition) is asked to enter their name.
- Next, the leader is asked to enter the names of the other four travelling companions
- Player starts in Independence Missouri (mile 0).
- There are 2,040 miles to go until the arrival in Oregon City.
- Player starts with $1200.
- Before departing, the player spends $200 for purchasing a wagon.
- Before the start of the trip and after purchasing the wagon, the player can visit the store and buy supplies: food, oxen, bullets and wagon parts. More details on the “Visit the store” activity later.
- Player is given the choice to start on the default start date of 03/28/1847 or to choose a different departure date between 03/01 and 05/01.
    - Give the player the option to start on the default date
    - If they refuse, ask them to choose their starting month, followed by the starting day
- The player must arrive in Oregon City by November 30th, 1847.

Visit the Store:

At the beginning of the trip, the player has the opportunity to purchase things they might need on their long journey. Below is a modified excerpt from the original game, which you can also find in the file *“store_info.txt”*. (If you’re following along with the 1985 video, you will notice the store information is presented in a slightly different manner, through the store owner, Matt.)

```
YOU HAD SAVED $1200 TO SPEND FOR THE TRIP, AND YOU'VE JUST PAID $200
FOR A WAGON. YOU WILL NEED TO SPEND THE REST OF YOUR MONEY ON THE
FOLLOWING ITEMS:
```

```
-   OXEN. YOU CAN SPEND $100-$200 ON YOUR TEAM. THE MORE YOU SPEND,
THE FASTER YOU'LL GO BECAUSE YOU'LL HAVE BETTER ANIMALS
-   FOOD. THE MORE YOU HAVE, THE LESS CHANCE THERE IS OF GETTING SICK
-   AMMUNITION - YOU WILL NEED BULLETS FOR ATTACKS BY ANIMALS AND
BANDITS, AND FOR HUNTING FOOD
-   MISCELLANEOUS SUPPLIES. THIS INCLUDES MEDICINE AND OTHER THINGS
YOU WILL NEED FOR SICKNESS AND EMERGENCY REPAIRS
YOU CAN SPEND ALL YOUR MONEY BEFORE YOU START YOUR TRIP, OR YOU CAN
SAVE SOME OF YOUR CASH TO SPEND AT FORTS ALONG THE WAY WHEN YOU RUN
LOW. HOWEVER, ITEMS COST MORE AT THE FORTS. YOU CAN ALSO GO HUNTING
ALONG THE WAY TO GET MORE FOOD.
```

Present the user with the information about possible store purchases. You can use some of the text in the fragment above or you can modify it to include more information or to look more like the 1985 version from the video.

Then, ask the user if they wish to purchase things in the following 4 categories:

1. Oxen:
   a. The user must spend between $100 and $200 dollars on oxen, inclusive.
   b. There are 2 oxen in a yoke and the price of each yoke is $40. Present the oxen price information to the user and ask how many yokes do they wish to purchase.
   c. If the total sum spent on oxen is not between $100 and $200 dollars, ask the user to choose again.
   d. After the purchase is in the satisfactory range, print out the current bill so far.

2. Food:
   a. The store owner recommends the player(s) should purchase at least 200 lbs. Of food per person, at 50 cents per pound.
   b. Ask the user to how many pounds of food they wish to purchase.
   c. Compute and print the current bill so far.

3. Bullets:
   a. A box of 20 bullets costs $2.
   b. Ask the user to how many boxes they wish to purchase.
   c. Compute and print the current bill so far.

4. Miscellaneous Supplies:
   a. A wagon part (wheels, axles, tongues) costs $20.
   b. Ask the user to how many parts they wish to purchase.
   c. A medical aid kit costs $25.
   d. Ask the user to how many kits they wish to purchase.
   e. Compute and print the current bill so far.

Note1: At any time in the buying process, if the player is attempting to buy a item (or a number of items of the same kind) whose price exceeds the total cash the player has in hand, print a message to the player informing them they do not have enough money and ask them to choose another quantity of the item in question.

Note2: The player can choose to purchase items in a certain category multiple times during the same store visit. For example, the player might choose the purchase oxen, followed by food, then oxen again, then more food, then bullets but no medical supplies or wagon parts. After every new set of items added, your program should print the bill so far.

Other stores will be available along the trail, at some of the forts along the way. The player will be given the opportunity to visit these stores and replenish their supplies. The prices at the stores along the trail get progressively higher as you get further along. At the first store, the prices will be **25% higher** that the ones at the store in Independence, Missouri. At the second store, they will be **50% higher**, at the third store **75% higher**, and so on.

Taking turns:

- In general, each turn represents 2 weeks of time. You typically travel between 70 and 140 miles each turn. If the player chooses to rest or hunt, the turn will not take 2 weeks. Entire trip could take 20 turns or more.
- At the beginning of each turn (and also before the start of the trip), your program needs to display a **Status Update**. Here are the categories you need to display:
    - Current date (mm-dd-yyyy)
    - Miles traveled (in miles from the start of the trip)
    - Next landmark (in miles)
    - Food available(in lbs.)
    - Number of Bullets available (an integer)
    - Cash available (in $)

Note: A screenshot of the Status Update, as it appeared on the original Oregon Trail Game, is supplied with the project files (*status_update.pdf*). Use it as guidelines; please note that some of the items listed in the file do not apply to our project.

Each turn, the player is asked to choose between the following actions:

1. Stop to **rest**:
    a. Resting takes between 1 and 3 days (a turn where the player chooses to rest does not take 2 weeks)
    b. The travelling party consumes 3 lbs. of food, per person, per day
    c. If any members of the travelling party are sick, they can recover at full health in two days

2. **Continue** on the trail:
   a. A turn where the player chooses to continue on the trail takes 2 weeks
   b. The travelling party consumes 3 lbs of food, per person, per day
   c. You typically travel between 70 and 140 miles.

3. **Hunt:**
   a. Hunting takes 1 day (a turn where the player chooses to hunt does not take 2 weeks)
   b. The player could encounter a rabbit, a fox, a deer, a bear or a moose, with the following probabilities:
      - rabbit: 50%
      - fox: 25%
      - deer: 15%
      - bear: 7%`
      - moose: 5%

   c. Figure out, in order, if the player has encountered each of the animals specified above (this means the player could encounter no animal, just one animal, ..., or they could encounter all 5 animals on the hunting trip)
   d. Present the player with the hunting opportunity, specifying which animal they have encountered. Ask the player if they want to hunt or not.

```
YOU GOT LUCKY! YOU ENCOUNTERED A DEER! DO YOU WANT TO
HUNT: (1) YES, (2) NO
```

   e. If the player chooses to hunt, but they have less than 10 bullets left, then the hunt is unsuccessful.
   f. If the player chooses to hunt and they have more than 10 bullets left, they must solve a puzzle (see the puzzle section later in the project description)
   g. If the travelling party has a successful hunt, they will improve their total food as follows:
      - rabbit: 2 lbs.
      - fox: 5 lbs.
      - deer: between 30-55 lbs.
      - bear: between 100-250 lbs.
      - moose: between 300-600 lbs.

   h. If the travelling party has a successful hunt, they will lose some of their ammunition, as follows:
      - rabbit: 10 bullets.
      - fox: 8 bullets.
      - deer: 5 bullets.
      - bear: 10 bullets.
      - moose: 12 bullets.

   i. At the end of the hunting day, the are asked how well do they want to eat. Note: this choice only happens at the end of the day when the player chooses to hunt:

- Poorly: 2 lbs of food, per person
- Moderately: 3 lbs of food, per person
- Well: 5 lbs of food, per person

j. After they finished eating, you can compute the total lbs. of food the travelling party has acquired through hunting. Even with a wildly successful hunting day, the wagon cannot hold more than 1000 lbs. of food. If the total food exceeds 1000 lbs., cut the food at 1000 lbs. and print a message explaining they had to leave the rest of the food behind.

k. If any members of the travelling party are sick, they can recover at full health during the hunting day if they can eat “well” at the end of the hunting day.

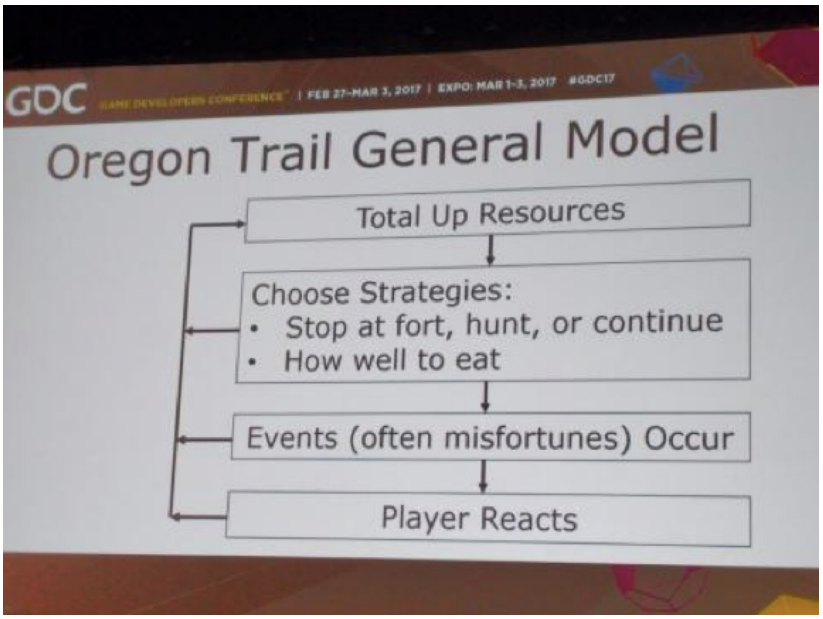
4. **Quit:** If the player chooses to quit, the game ends. Print a message expressing regret the travelling party had to cut their trip short.

Note: as the players advance in their travels, the date changes. You will need to keep count of months that have 30 days, vs 31 days or 28 days. Our suggestion: you might want to create a class to handle the date (day-month-year).

Misfortunes:

There is more to the game that resting, travelling and hunting. At every turn in the game, after the player chooses and completes one activity (resting, travelling, hunting), there is a **40% probability** a misfortunate event might occur.

Included below are the flow charts from the original Oregon Trail game. We will attempt to implement a pretty close version in this project.



If a misfortune occurs at the end of a particular turn, choose one misfortune, at random, between the following options:

1. One member of the travelling party gets sick. The possible diseases are: typhoid, cholera, diarrhea, measles, dysentery (of course), and fever. Print a message announcing the misfortune:

```
OH NO! JANE HAS THE MEASLES!
```

   - If that particularly member of the travelling party was already sick, getting sick again will make them lose their life. If the leader of the travelling party dies, the game ends.
   - Resting for two days will bring one's health back to 100%.
   - Hunting (1 day) and eating "well" will bring one's health back to 100%.
   - If they are not resting, they will be back to full health in 5 days.
   - If they travelling party has a medical aid kit, it will shorten their recovery time from 5 days to 2 days

2. One of the oxen dies. Print a message announcing the misfortune:

```
OH NO! ONE OF THE OXEN HAS DIED! YOU HAVE … OXEN LEFT.
```

   - The travelling party will become unable to continue their journey if all their oxen die

3. A thief attacks during the night. Print a message announcing the misfortune:

```
OH NO! LAST NIGHT YOU GOT ATTACKED BY A THIEF!
```

   - The thief will steal between 10-25 lbs. of food.

4. One of the wagon wheels/axles/tongues breaks. Print a message announcing the misfortune:

```
OH NO! ONE OF YOUR WHEELS IS BROKEN!
```

   - If the player has spare wagon parts, it takes 1 day to fix the wagon before a new turn starts. The number of spare Parts goes down by 1. The food will go down at the daily rate. The health of the travelling members will improve.
   - If the player does not have spare wagon parts, the travelling party will become unable to continue their journey

5. Bad weather: heavy rains, storm, hail, blizzard or hurricane. Print a message announcing the misfortune:

```
OH NO! THE WEATHER IS ROTTEN! HAIL!
```

- The party is forced to rest until the calamity passes: 1 day for heavy rain and hail, 3 days for storm and blizzard, and 7 days for hurricane. The food will go down at the daily rate. The health of the travelling members will improve.

Note: If you want to implement "positive events" you are free to include those as well. For example: `"The family discovered a lost pirate treasure chest. Your cash reserves increase by $1000".`

Raider Attack:

No two trips to Oregon were exactly the same. Players came into contact with people they didn't know on the trail. It was more likely that you would meet with strangers at the beginning of the trip instead of the end because you a) weren't so far from civilization so people could rob you and get back, and b) the geography was more flat and easy to survive than in the mountains.

At the end of every turn, and after some unfortunate event might occur, there is a probability the player will be attacked by raiders. The original formula used to calculate the probability of being attacked by raiders was based on mileage traveled, *M*:

$$\textbf{Probability of meeting raiders} = \frac{((M/100-4)^2+72)}{((M/100-4)^2+12)} - 1$$

If, using the probability calculated using the formula above, the riders attack occurs, print a message announcing the event:

```
RIDERS AHEAD! THEY LOOK HOSTILE!
```

The player will then be given three choices:

1. **Run:**
   - The travelling party escapes with their lives but, in the hurry to flee the raiders, they left behind 1 ox, 10 lbs. of food and 1 wagon part.

2. **Attack:**
    - In order to win the battle, the player must pass a puzzle (see the puzzle section later in the project description).
    - If they win the battle, the travelling party gains 50 lbs. of food and 50 bullets.
    - If they lose the battle, the travelling party loses a quarter of their cash reserves and 50 bullets.

3. **Surrender:**
    - The travelling party loses a quarter of their cash reserves.

Puzzles:

The player has to pass a puzzle in order to win the hunt, or in order to fight off the raiders. Every time a puzzle is required, the program must generate a random number between 1 and 10. The player has 3 tries to guess the number. If the player guesses correctly, they solve the puzzle.

Milestones:

All the milestones are written in a file "*milestones.txt*". Here is the content of the file:

```
Kansas River Crossing
      102mi River 4ft
Big Blue River Crossing
      185mi River 3ft
Fort Kearney
      304mi Fort
Chimney Rock
      554mi
Fort Laramie
      640mi Fort
Independence Rock
      830mi
South Pass
      932mi
Fort Bridger
      989mi Fort
Green River Crossing
      1151mi River 5ft
Soda Springs
      1295mi
```

```
Fort Hall
      1395mi Fort
Snake River Crossing
      1534mi River 5ft
Fort Boise
      1648mi Fort
Blue Mountains
      1808mi
Fort Walla Walla
      1863mi Fort
```

Important considerations:

1. Some milestones are forts, some are river crossings and some are just landmarks along the way. You can figure out what type of milestone it is by reading the file and counting the number of words on each line:

Example 1:

```
Kansas River Crossing
102mi River 4ft
```

The milestone is a **river** because the second line has 3 words in it. The last word value represents the river's depth.

Example 2:

```
Fort Kearney
304mi Fort
```

The milestone is a **fort** because the second line has 2 words in it.

Example 3:

```
Chimney Rock
554mi
```

The milestone is a **landmark** because the second line has 1 word in it.

2. As the player randomly generates the length traveled in a 2-week cycle, compare the total mileage with the mileposts of the next milestone. If the total travel would exceed the next milestone, adjust the traveled distance down in order to stop at the next milestone.

Example:

```
MILES TRAVELED: 0
NEXT MILESTONE: 102mi
...
```

```
YOU WERE PREPARED TO TRAVEL 120 MILES BUT YOU ARRIVED AT THE KANSAS
RIVER CROSSING. WHAT DO YOU WANT TO DO:
(1) REST, (2) CONTINUE
2

MILES TRAVELED: 102
NEXT MILESTONE: 83mi
...
...
```

3. Here are the game options, for every type of landmark:
    - At a landmark, the player can choose to rest (repeatedly) or to continue.
    - At a river, the player can choose to rest (repeatedly) or to cross the river. Any river deeper than 3 feet will require a ferry over the river. The ferry costs $10 per person. After crossing, the journey continues with a new turn.
    - At a fort, the player can choose to rest (repeatedly), visit the store (repeatedly), or continue the journey.

4. The last milestone is Oregon City, 2040 miles from the beginning and 177 miles from Fort Walla Walla.

Game end:

The player wins the game when the party (not necessarily all the members) reaches the destination before the deadline.

The player loses the game when:

- The party runs out of food.
- The party has lost all the oxen.
- The party has a broken wagon and has no wagon parts left
- The first player (the leader) dies. Note: If other family members die, the leader can still push ahead, reach the destination and win the game.

If the game ends, your program should print a message informing of the tragic event.

```
YOU HAVE DIED OF DYSENTERY!
```

, along with the following: leader name, miles travelled, food remaining, cash remaining.

Note: the final stats of the game should also be saved in a file named *results.txt*

Minimum Requirements:

Your implementation of the Oregon Trail should have:

- 3+ user defined classes
- At least two classes with 4+ data members.
    - At least one class must include an array of objects from a class that you created.
- Appropriate methods for each class (including getters and setters)
- Your project implementation must include at least:
    - 4+ if / if-else statements
    - 4+ while loops
    - 4+ for loops
    - 2+ nested loops
    - 4+ strings variables/data members
    - File I/O for reading (from the provided text files)
    - File I/O for writing some data members of an object (save to *results.txt*)
- Your project must have an interactive component (ask the user for input, create a menu for choices, ...).

What do you need to submit:

1. **Checkpoint 1: Choose Project (due Monday, March 19th by 6 pm)**
   If you choose to implement the Oregon Trail project, you must submit a text file (.pdf) informing your TA of your choice:

   ```
   John Doe
   Section #107
   TA name: …

   I am choosing Oregon Trail for my Project
   ```

   Failure to submit your project choice will result in a 5/100 penalty on your Project III score.

   Note: if you choose to create your own project, follow the instructions for the Project Proposal in the *ChooseYourOwn* Project write-up.

2. **Checkpoint 2: Class files & Code Skeleton (due Sunday, April 8th by 6 pm)**

For this checkpoint, you will need to complete your class files and create your driver files.

Your .h files should be *complete* with all the data members and member functions you will be using for each class. For the class implementation .cpp files, you should fully implement simple functions like your getters and setters. For more complex functions you can include function stubs with detailed comments.
For example, if I were stubbing a function to implement bubble sort and return the number of swaps I might do:

```
/*
   1. Compare adjacent elements. If the first is greater than the
      second, swap them.
   2. Do this for each pair of adjacent elements, starting with
      the first two and ending with the last two. At this point
      the last element should be the greatest.
   3. Repeat the steps for all elements except the last one.
   4. Repeat again from the beginning for one less element each
      time, until there are no more pairs to compare.
*/
int bubble_sort(int arr[], int size)
{
      int swaps = 5;
      return swaps; // function returns expected type (int)

}
```

The driver file should contain detailed comments with pseudocode that explains the functionality of the project. (It should look similar to the libraryDriver.cpp you submitted for Homework 7 Part I).

You must submit your Class Files and Code Skeleton to Moodle to get full credit for the assignment. The penalty for not submitting the Class Files and Code Skeleton is 10 points. Zip all your .h and .cpp files and submit the archive file on Moodle by Sunday, April 8th by 6 pm.

Failure to submit the Class Files and Code Skeleton will result in a 10/100 penalty on your Project score.

3. Checkpoint 3: Final Deliverables (due Sunday, April 22nd by 6 pm)

The final version of your project will be due on Sunday, April 22nd by 6 pm (no bonus available). You must submit a .zip file to Moodle which includes:

- All .h and .cpp files including the main driver program, correctly indented and commented.

4. **Checkpoint 4: Project Report - Reflection Activity (due Sunday, April 29th by 6 pm)**

   Reflection questions:

   - How did you prepare for the Project?
   - Did you write a Code Skeleton? Was it useful? How?
   - Reflect on how you could have done better, or how you could have completed the project faster or more efficiently.

   What you need to do:

   Write a 1-2 page report containing answers to the reflection questions.

   In addition, write a paragraph answering the following question, in the context of the Project in CSCI 1300:

   Did you have any false starts, or begin down a path only to have to turn back when figuring out the strategy/algorithm for your Final Project program. Describe in detail what happened, for example, what specific decision led you to the false starts, or, if not, why do you think your work had progressed so smoothly, and give a specific example.

   Note: all reflection papers should be individual.

   Upload your report as a pdf on Moodle under *Final Project Report* before Sunday, April 29th by 6 pm.

   Failure to submit the Report will result in a 5/100 penalty on your Project score.

Collaboration:

All code written for this assignment must be your own. You *may* work together to come up with project ideas and to work through errors. Make sure to give credit to those you work with!

You *may not* use code provided or taken from anyone or anywhere else. **All code must be your own.** In particular, your projects should have different classes, different implementations, and different behavior.

If you use any resources (web or otherwise) you will need to acknowledge them at the top of your main program.

Example:

```
/*sources:
1.
https://stackoverflow.com/questions/9622163/save-plot-to-image-file-instead-of
-displaying-it-using-matplotlib
#used to find examples for how to save a plots vs showing a plots

2.
https://stackoverflow.com/questions/1557571/how-do-i-get-time-of-a-python-prog
rams-execution
#used to learn how to print execution time

3.
I worked with Jordan to brainstorm ideas for the project

4.
My TA helped my fix some of my errors with the function that writes to file
*/
```

You cannot copy all or most of your code from a resource, acknowledge it, and receive points for this project. If your code is very similar to an online resource or to one of your classmates or to one of the students who took this class (or another intro programming) before, it will receive a 0 and will guarantee you a meeting with the Honor Code, regardless if you acknowledge your source or not. You must show that you substantially contributed to the project, that your project does not contain just parts copied from other sources. If you have questions about what this means, please come speak to an instructor.

Interview grading for the project will begin on **April 23rd**, the Monday after the submission deadline.

Points:

If your code does not compile, you can get a maximum of 40 points for the project.

40 points for interview grading

- TA's questions about your project
- code compiles
- algorithms descriptions, comments, good style

20 points for minimum requirements(this is the *maximum* you can get if your code does not compile)

- code meets the minimum requirements specified on the first page (number of classes, loops, etc.)

40 points for project functionality

Possible Deductions:

- 5 points for not submitting a proposal (or your choice for Oregon Trail)
- 10 points for not submitting code skeleton and class files
- 5 points for not submitting report