Lab 1: Review of Computing Principles
CMSC 162
Spring 2025
Due: Feb. 10th, 11:59:59pm

This lab has three parts. First, I have given you a series of review tasks. Second, I have given you a short simulation project to implement. Finally, you will need to do "glitter" (in which you extend the project in a creative and unique way).

I have provided you with some starter code, which you can download with this wget command:

wget https://marmorstein.org/~robert/Spring2025/cs162/Lab1-files.tar.gz

Then extract them with:

tar xvf Lab1-files.tar.gz

This will create a Lab1/ folder which contains several files. For each of the five tasks, I've provided you with a .cpp file (for example, the task1.cpp file is for Task 1). You will need to edit these files to solve each task.

There are also two directories: Dreidel/ and Glitter/ which contain code for the final two parts of the project.

Makefiles

A Makefile is a sort of recipe for compiling a program that can be used by the Unix "make" command to build a project. When a programming projects is very large and involves several different files or when compiling requires passing many different flags to the compiler, it can be very convenient to use a Makefile instead of typing the "g++" command by hand over and over.

I have provided you with a Makefile in the Lab1 folder. To use it, you can type:

cd Lab1/
make

From now on, you can compile your project by typing "make" and pressing enter. You can compile a specific task by typing "make Task1" (for example).

Task 1. Write a complete C++ program that takes command line arguments and prints out the last argument.

For example, if I ran the program like this:

```
./Task1 This is a test of the emergency alert system.  It is only a test.
```

The program would print:

```
test.
```

Task 2. Write a program which reads in a string of 1's and 0's (i.e. a binary number) from the keyboard and outputs the decimal value of that number.

For example, if I run the program and then type: 00000110

The program would print: 6

If I type: 01100001

The program would print: 97

If I typed: 1111

The program would print: 15

Your program should be able to handle any number 1 to 32 bits long. You may assume that all numbers are (positive) unsigned integers.

Task 3. Write a program that reads numbers from standard input until it encounters an EOF (end of file). Then prints the sum of those numbers.

Numbers can be either positive or negative, but neither the numbers or the sum will ever exceed two billion or be less than negative two billion. **CTRL-D** means to hold down control on your keyboard while pressing the key for the letter 'd'. It is a way to send an "End of File" signal from the keyboard.

For instance, if I run the program and then type:

```
45
10
15
30
CTRL-D
```

The program would print: 100. If I typed:

```
20
-5
3
7
10
CTRL-D
```

The program would print: 35. If I typed:

```
-100
10
CTRL-D
```

The program would print: -90

Task 4. Write a program which reads in six floating point values from the standard input, sorts them, and prints them back out. When printing, each value should be printed on its own line with a precision of two decimal places.

For example, if I run the program and then type:

```
3.4 10.2 1 0 -0.0 -5.3
```

The program should print:

```
-5.30
0.00
-0.00
1.00
3.40
10.20
```

Task 5. Write a program that reads in a line of text from the keyboard, shifts every letter (but only letters) to the next letter of the alphabet (z wrapping back around to a), and prints it back out.

For example, if I run the program and then type:

```
The fat zebras are really blue today, but the horses are green.
```

The program would print:

```
Uif gbu afcsbt bsf sfbmmz cmvf upebz, cvu uif ipstft bsf hsffo.
```

Notice that neither spaces nor punctuation change – only letters. Also, your program should preserve the case of the letter.

Hints: To do this, you will probably want to use:

1. A string (to store the message).
2. The getline function (to read in the string)
3. A loop (to go through each character of the message)
4. An if statement (to check if the current character is a letter)
5. A cout statement (to print out the shifted letter)

Because characters are represented as ASCII codes (binary numbers) in the computer, you can use operations like + and – on them. To shift a letter, you can just "add one" to it.
Of course, if you do that to "z" you'll get "{" instead of "a" and if you do that to "Z" you'll get "[", so you may want to use else-if to handle that case. Alternatively, you can use modular arithmetic. Either solution is acceptable.

The Dreidel Simulation

The second part of this lab is to write a simulation of the Dreidel game, a gambling game played by children to celebrate several Jewish holidays (but particularly Hanukkah).

Each child starts the game with a certain number of “geld” (special chocolate coins – sometimes covered in gold foil). Each child puts one piece of geld into the “pot”. The children then take turns spinning the dreidel, a four-sided top labeled with Hebrew letters.

Depending on how the Dreidel lands, the following occur:

1. If a “Nun” is spun, nothing happens.

2. If a “Shin” is spun, the child who spun the dreidel has to put one additional geld into the pot.

3. If a “He” is spun, the child who spun the dreidel gets half the pot.

4. If a “Gimel” is spun, the child who spun the dreidel gets the entire contents of the pot. Since this empties the pot, every child still playing then puts one geld into the pot (including the child who spun the dreidel).

If a player runs out of geld, they are out of the game. Play continues until only one child is left (at least in our version).

I would like you to write a computer simulation that simulates the game for different numbers of players and different starting amounts of geld and counts how many rounds it takes for the game to end.

Input:

The syntax of your dreidel command should be:

./Dreidel <number_of_players> <number_of_starting_coins>

For example, I should be able to type: ./Dreidel 4 20

To see what happens when four players start with 20 geld.

Output:

Every time your program rolls the dreidel, it should print out the Hebrew letter that comes up on that roll. Then, on a separate line, it should print the word “Pot:” followed by a space, the number of geld in the pot, the word “Players: “, and then a space separated list of numbers showing how much geld each player currently has.

At the end of the simulation, your program should print the words “Total Rounds: “ followed by an integer giving the total number of rounds the simulation took.

For example, if I run the simulation with the command: “./Dreidel 2 3”, The program should print something that looks very much like:

Nun
Pot: 2 Players: 2 2
Gimel

Pot: 2 Players: 1 3
Gimel
Pot: 2 Players: 2 2
He
Pot: 1 Players: 2 3
Gimel
Pot: 2 Players: 2 2
He
Pot: 1 Players: 2 3
Gimel
Pot: 2 Players: 2 2
He
Pot: 1 Players: 2 3
Gimel
Pot: 2 Players: 2 2
Gimel
Pot: 2 Players: 1 3
Nun
Pot: 2 Players: 1 3
Nun
Pot: 2 Players: 1 3
He
Pot: 1 Players: 2 3
Shin
Pot: 2 Players: 2 2
Shin
Pot: 3 Players: 1 2
Nun
Pot: 3 Players: 1 2
Nun
Pot: 3 Players: 1 2
Nun
Pot: 3 Players: 1 2
He
Pot: 2 Players: 2 2
Nun
Pot: 2 Players: 2 2
He
Pot: 1 Players: 3 2
Nun
Pot: 1 Players: 3 2
Shin
Pot: 2 Players: 2 2
Shin
Pot: 3 Players: 2 1
Nun
Pot: 3 Players: 2 1
Gimel
Pot: 2 Players: 1 3

Nun
Pot: 2 Players: 1 3
He
Pot: 1 Players: 1 4
He
Pot: 1 Players: 1 4
Gimel
Pot: 2 Players: 0 4
Total rounds: 15

Random Numbers
In order to ensure consistent grading, set the random number generator seed to 20 by placing the following command at the beginning of your main function:

srandom(20);

Hints:

1. I recommend using a vector of integers to store the amount of gold each player has and single integers to represent the number of players and the amount of geld in the pot.

2. In your main function, you can loop "num_player" times and use "push_back" to give each player the correct amount of starting geld. Then you can use a while loop to simulate repeated spins of the dreidel.

4. To convert a C-string (such as argv[1]) to an integer, you can use the "atoi" function.

5. It helps to break your program up into functions. In my implementation, I had the following functions (in addition to "main"):

```
void ante(); //Looped through each player, decreased their geld by one and increased the pot by
                the same amount.
int players_alive(); // Looped through the geld vector to see count how many players still had
                        geld left and returned the total.

void spin_dreidel(int player); //Simulated one spin of the dreidel by player "player".
```

Glitter

In the Glitter/glitter.cpp file, extend the project in a creative way. To help you, I've provided some code in the Dreidel folder you may want to copy over and use. The functions of the "Dreidel" class draw representations of each side of a dreidel. If you like, you can use them, but you can also do something completely different. It's up to you.

As usual, I will grade your glitter on both its technical and artistic creativity. Glitter that clearly took time to produce will earn more points than glitter that only took a few lines of code and little planning.

Glitter is NOT extra credit. It is a required part of this project that demonstrates that you have mastered the material well enough to extend it.

Submitting

If you are in the Lab1 folder, change directories to the parent folder:

```
cd ..
```

Then create a tarball of your Lab1 folder like this:

```
tar czpvf Lab1.tar.gz Lab1/
```

If you've done everything correctly, Linux should give you a list of files that looks something like this:

```
Lab1/
Lab1/Dreidel/dreidel.cpp
Lab1/Dreidel/dreidel.h
Lab1/Dreidel/main.cpp
Lab1/Glitter/glitter.cpp
Lab1/Makefile
Lab1/task1.cpp
Lab1/task2.cpp
Lab1/task3.cpp
Lab1/task4.cpp
Lab1/task5.cpp
```

There may be other files as well, but that's fine.

Then use a web browser to connect to the course web site at

```
http://marmorstein.org/~robert/submit/
```

Log in using your submit system credentials and upload the **`Lab1.tar.gz`** file to the course web site.

Be careful! The **Lab1-files.tar.gz** archive is also in your home directory and is NOT what you want to submit.