Assignment 1

The purpose of this assignment is to make sure your programming environment is set up correctly. That is, you have an editor / IDE, a compiler, and the make build system installed on your computer. It is also a chance to work with git and github and an opportunity to write something for the **production environment**.

Question 1

Implement a simple "day of the week" program in C++. Your program should output the day of the week based on the input according to the mapping below. For any other number or value not in the table below, your program should output "`Invalid Value!`"

Exit / Return Codes: Your program should finish with the correct exit code depending on the user input. If the user inputs a valid input number, the program should exit with code 0. If the user inputs an invalid number (e.g., `-2` or `16`) the program should exit with error code 1. The code can be passed when calling `return` in the main function, or by calling `exit(code_num)` at any time.

Makefile: Your program should have a `Makefile` containing an "`all`" directive that compiles the program and outputs an executable named `dow` (day of week).

Snark: If the user enters a really large number, that is a clue (to the programmer) that they have no idea how this program works. Modify your program so that if the user enters a value smaller than `SHRT_MIN` or larger than `SHRT_MAX` it insults them and then gives them some guidance. See the sample output below. In this case the `exit code` should be 2.

Input Number	Output
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

Sample Output:

```
user@machine$ make
g++ -Wall -g dow.cpp -o dow
user@machine$ ./dow
Number: 2
Tuesday
```

```
user@machine$ $?
0
user@machine$ ./dow
Number: -2
Invalid Value!
user@machine$ $?
1
user@machine$ ./dow
Number: 7234235
7234235?!!1!?! Are you an idiot or something?
The only valid inputs are 0 - 6.
user@machine$ $?
2
user@machine$
```

Bonus Question (ungraded): You may have noticed that you can input a letter (instead of a number) which causes some odd results. Can you explain what is happening?

Question 2

Many students come into this class with the preconceived notion that it's, "really hard." The first part of this assignment might have been a let-down in that regard. Well don't worry! It will definitely get harder! But you can do it. Here is another problem that might do a better job of living up to your lofty expectations.

You are working at a mythical palace of balance and equilibrium. The monks (you are a monk now) at the palace pride themselves on finding perfectly diplomatic, fair, optimal, and EQUAL solutions to problems of all kinds. They do a lot of work in political and diplomatic disputes. One day a duczoll alien from a far away planet (why not) came to the palace with a balance problem involving a rare mineral found on their home planet called "ocus." Ocus is used on the doczoll home world to provide energy. Ocus pieces are much more valuable if there are several together, all of equal size. If they are all equal size the energy they can produce in increased by a factor of 10-thousand. Far more than the energy production of each alone or as an unbalanced set.

Problem Statement: There are several pieces of ocus that need to be of equal size. Piece one is $1cm^3$, piece number two is $1cm^3$, and piece number three is $5cm^3$. The duczoll have a machine that can grow the ocus pieces, but only in a very particular way. Before the machine is run it must be set to grow the ocus pieces by $1cm^3$ **or** $2cm^3$ **or** $5cm^3$. The growth value selected will then be applied to all of the ocus in the machine except for the largest one (i.e., the largest ocus piece before running the machine), due to a mysterious property the doczoll call "the active matrix effect." The machine can be run many times, but each run is very expensive taking approximately 2 years.

The problem of growing the ocus stumped the monks. Can you solve it? Can you minimize the number of runs necessary to make all the ocus pieces equal size?

In a text file "`machine_settings.txt`" in your git repository answer the following questions:
(a) How many runs of the machine are necessary?

(b) What are the settings of the machine on each run?
(c) What are the sizes of the ocus pieces before / after each run?

Bonus Question (ungraded): Suppose the duczoll had a different set of ocus pieces they needed to balance in size. Would your solution still be correct? Is there a generic way to compute how to configure and run the machine?

Submission: Your program will be submitted using canvas and git + github.

1) You should create a private `github.com` repository (repo) with your code in it. You may need to create a github user account and a github repository. To achieve this. The name you choose for the repo and your username are arbitrary but I suggest "HW1" as the repo name.

2) Your repository on github.com should be **private** and you should add my account as a collaborator by going to "settings" → "manage access" → "invite collaborator" and entering my username: `fmresearchnovak`

3) On canvas you should upload a link to this github repository
`https://github.com/fmresearchnovak/HW1.git`

I will grade the final commit made before the due-date. This means that you can actually submit on canvas at any time before the deadline, and still make changes (new commits) to the code.