

Galactic Warfare Simulation

you should always use `std::priority_queue<>`

Galaxy Logic Overview

Your program, named `galaxy`, will receive as input a series of “**battalion deployments**,” or placements of troops on a certain planet. A **battalion deployment** consists of the following information:

- Timestamp — the time that this deployment order is issued.
- General ID — the general who is issuing the deployment order.
- Planet ID — the planet which the troops are being deployed to.
- Jedi or Sith — Whether the General issuing the deployment is a Jedi or Sith.
(Flavor note: in Star Wars, the Jedi are the good guys on the Light side of the Force, and the Sith are the bad guys on the Dark side of the Force. Click the link below for more info on the Force).
- [Force-sensitivity](#) — the average Force-sensitivity of the troops being deployed.
- Quantity — the number of troops being deployed.

As you read each battalion deployment from input, your program should see if the new battalion can be **matched** with a battalion previously deployed on the planet. If a match occurs, then the two battalions engage in warfare. A new battalion can be matched with a previously deployed battalion if:

- Both deployments are for the same planet.
 - General ID does not matter; generals are allowed to switch sides of the Force: i.e. order conflicting battalion deployments.
- The deployments were issued on different sides of the Force. That is, one deployment was a Jedi deployment and the other was a Sith deployment.
- The Jedi Force-sensitivity is **less than or equal** to the Sith Force-sensitivity.
 - The Sith always instigate fights, and they only fight battalions with Force-sensitivity which they are confident that they can overcome.
 - This does not mean that a battle will not occur if a Jedi battalion is being deployed. When a Jedi battalion is deployed, they may be ambushed by a previously deployed Sith battalion.

If the new battalion is a Sith battalion, it will **always** choose to attack the least Force-sensitive Jedi battalion on the planet, given that there is a Jedi battalion with lesser Force-sensitivity. If the new battalion is a Jedi battalion, it will **always** be ambushed by the most Force-sensitive Sith battalion on

the planet, given that there is a Sith battalion with greater Force-sensitivity. In the event of a tie in Force-sensitivity, choose the battalion that was deployed first (came first in the input file).

When a battle occurs, the battalions trade troops one-for-one, regardless of their Force-sensitivity. That is to say that an equal number of troops from both battalions are eliminated, equal to the number of troops in the smaller battalion. If one of the battalions survives, it remains on the planet for future possible fights. For example, if a Sith battalion with 20 troops fights a Jedi battalion of 30 troops, the Sith battalion is eradicated and the Jedi battalion remains with 10 troops.

In the event that the newly deployed battalion survives, it is possible that a new fight will break out. If the new battalion is Sith, they will then look to attack another Jedi battalion. If the new battalion is Jedi, they may be attacked again after defeating the first Sith battalion. This happens until no more fights break out: that is, there are no pairs of Jedi and Sith battalions remaining on the planet such that the Sith Force-sensitivity is greater than or equal to the Jedi Force-sensitivity.

Input

Input will arrive from standard input (`cin`). There are two input formats, *deployment list* (DL) and *pseudorandom* (PR). The first four lines of input will always be in the following format, regardless of input format, which you may assume are correctly formatted:

```
COMMENT: <COMMENT>
MODE: <INPUT_MODE>
NUM_GENERALS: <NUM_GENERALS>
NUM_PLANETS: <NUM_PLANETS>
```

<COMMENT> is a string terminated by a newline, which should be ignored. (You should comment your test files to explain their purpose.)

<INPUT_MODE> will either be the string "DL" or "PR". DL indicates that the rest of input will be in the deployment list format, and PR indicates that the rest of input will be in pseudo-random format. Details for these input formats will be explained shortly.

<NUM_GENERALS> and <NUM_PLANETS>, respectively, will tell you how many generals and planets will exist.

Deployment List (DL) Input:

In Deployment List mode, the rest of the input will be a series of lines in the following format:

```
<TIMESTAMP> <SITH/JEDI> G<GENERAL_ID> P<PLANET_NUM> F<FORCE_SENSITIVITY> #<NUM_TROOPS>
```

Each line represents a unique deployment. For example, the line:

```
0 JEDI G0 P1 F100 #44
```

Can be translated as:

“At timestamp 0, Jedi general 0 deploys 44 Jedi troops with Force-sensitivity 100 to planet 1.”

DL Input Error Checking

To detect corrupt deployment orders, you must check for each of the following:

- **<GENERAL_ID>** and **<PLANET_ID>** are integers in ranges $[0, \text{<NUM_GENERALS>})$ and $[0, \text{<NUM_PLANETS>})$, respectively.
 - e.g. if **<NUM_GENERALS>** is 5, then valid general IDs are 0, 1, 2, 3, 4.
- **<FORCE_SENSITIVITY>** and **<NUM_TROOPS>** are greater than 0.
- Timestamps are non-decreasing.
 - e.g. 0 cannot come after 1, but there can be multiple deployments with the same timestamp.

If you detect invalid input at any time during the program, print a helpful message to `cerr` and `exit(1)`.

You do not need to check for input errors not explicitly mentioned here.

Pseudorandom (PR) Input:

If **<INPUT_MODE>** is PR, the rest of input will consist of these three lines in this format:

RANDOM_SEED: **<SEED>**

NUM_DEPLOYMENTS: **<NUM_DEPLOYMENTS>**

ARRIVAL_RATE: **<ARRIVAL_RATE>**

- **RANDOM_SEED** — An integer used to initialize the random seed.
- **NUM_DEPLOYMENTS** — The number of deployment orders to generate. You may assume that this value will fit in an `int`.
- **ARRIVAL_RATE** — An integer corresponding to the average number of deployments per timestamp.

You may assume PR input will always be correctly formatted:

Generating deployments with P2random.h

We provide a file to generate the deployments in PR mode. This is to make pseudo-random generation uniform across platforms. The P2random class contains the following function:

```
void P2random::PR_init(std::stringstream &ss,          unsigned int seed,
                      unsigned int num_generals,      unsigned int num_planets,
                      unsigned int num_deployments, unsigned int arrival_rate);
```

`P2random::PR_init(...)` will fill the stringstream argument (`ss`) with deployments, so that you can use it just like you would `cin` for DL mode:

You may find the following C++ code helpful in reducing code duplication:

```

stringstream ss;

// inputMode is the "PR" or "DL" from line 2
if (inputMode == "PR") {
    // TODO: add code to read random seed, number of deployments, and arrival rate
    P2random::PR_init(ss, seed, num_gen, num_planets, num_deploys, rate);
} // if

// Create a reference variable that is ALWAYS used for reading input.
// If PR mode is on, refer to the stringstream. Otherwise, refer to cin.
// This is a place where the ternary operator must be used: an equivalent
// if/else is impossible because reference variables must be initialized
// when they are created.
istream &inputStream = inputMode == "PR" ? ss : cin;

// Make sure to read an entire deployment in the while statement
while (inputStream >> var1 >> var2 ...) {
    // Process this deployment, use PQs, make fights happen (if possible), etc.
} // while

```

DL & PR Comparison

The following two input files are in different modes, but should generate **the same deployments**.

COMMENT: DL mode generating some deployments.

MODE: DL

NUM_GENERALS: 3

NUM_PLANETS: 2

```

0 JEDI G0 P1 F100 #44
1 JEDI G0 P1 F56 #42
2 SITH G0 P1 F73 #19
2 SITH G0 P0 F34 #50
2 JEDI G0 P0 F86 #23
2 JEDI G0 P0 F20 #39
2 SITH G2 P0 F49 #24
2 JEDI G2 P0 F83 #45
3 JEDI G2 P1 F64 #22
3 JEDI G1 P0 F6 #19
3 JEDI G0 P1 F42 #37
4 JEDI G2 P0 F10 #44

```

COMMENT: PR mode generating the same sequence of deployments.

MODE: PR

NUM_GENERALS: 3

NUM_PLANETS: 2
RANDOM_SEED: 104
NUM_DEPLOYMENTS: 12
ARRIVAL_RATE: 10

Example Scenario

Consider the following series of deployments. The first example shows the format in which your program will take input in DL mode. The second example explains what each line of input means.

```
0 SITH G1 P2 F100 #10  
0 JEDI G2 P2 F10 #20  
0 SITH G3 P2 F1 #10
```

Sith general 1 deploys battalion 1 with Force-sensitivity 100 on Planet 2 with 10 troops.
Jedi general 2 deploys battalion 2 with Force-sensitivity 10 on Planet 2 with 20 troops.
Sith general 3 deploys battalion 3 with Force-sensitivity 1 on Planet 2 with 10 troops.

Here is a detailed explanation of what battles would occur as a result of the above input:

1. Sith battalion #1 lands on Planet 2 with 10 troops with Force-sensitivity 100.
 - There are no other battalions on the planet yet, so they remain on standby.
2. Jedi battalion #2 lands on Planet 2 with 20 troops with Force-sensitivity 10.
 - Sith battalion #1 encounters Jedi battalion #2. The Sith battalion #1 has a higher Force-sensitivity, so they instigate a fight with Jedi battalion #2.
 - They do battle and both lose 10 troops, since the troops are exchanged one-for-one.
 - Jedi battalion #2 remains on the planet, while Sith battalion #1 has been wiped out. Battalion #2 can battle at a later time with their remaining 10 troops.
3. Sith battalion #3 lands on Planet 2 with 10 troops with Force-sensitivity 1.
 - They do not engage Jedi battalion #2 because their Force-sensitivity is lower.
 - They remain on the planet and wait to attack a Jedi battalion with lower Force-sensitivity

Command Line Flags

Your program should take the following case-sensitive command-line options that will determine which types of output to generate. Details about each output mode are under the **Output Details** section.

- **-v, --verbose**
An optional flag that indicates verbose output should be generated.
- **-m, --median**
An optional flag that indicates median output should be generated.
- **-g, --general-eval**
An optional flag that indicates that the general evaluation output should be generated.
- **-w, --watcher**
An optional flag that indicates that movie watcher output should be generated.

Examples of legal command lines:

- `./galaxy < infile.txt > outfile.txt`
- `./galaxy --verbose --general-eval > outfile.txt`
- `./galaxy --verbose --median > outfile.txt`
- `./galaxy --watcher`
- `./galaxy --general-eval --verbose`
- `./galaxy -vmgw`

We will not be specifically error-checking your command-line handling; however we expect that your program conforms with the default behavior of `getopt_long()`. Incorrect command-line handling may lead to a variety of difficult-to-diagnose problems.

Output Details

The output generated by your program will depend on the command line options specified at runtime. With the exception of program startup and the end of program summary, all output is optional and should not be generated unless the corresponding command line flag is set.

Program Startup Output

Your program should always print the following line **before** reading any deployments:

Deploying troops...

Verbose Output

If and only if the `--verbose/-v` option is specified on the command line (see above), whenever a battle is completed you should print on a single line:

General `<SITH_GENERAL_NUM>`'s battalion attacked General `<JEDI_GENERAL_NUM>`'s battalion on planet `<PLANET_NUM>`. `<NUM_TROOPS_LOST>` troops were lost.

`<NUM_TROOPS_LOST>` is defined to be the total number of troops lost from both sides, or the number of Jedi troops lost + the number of Sith troops lost.

Example:

Given the following list of deployments:

```
0 JEDI G1 P0 F125 #10
0 SITH G2 P0 F1 #100
0 JEDI G3 P0 F100 #10
0 JEDI G4 P0 F80 #10
0 SITH G5 P0 F200 #4
```

No battles are possible until the **5th** battalion is deployed. When the 5th battalion is deployed and a

battle occurs, you should print:

General 5's battalion attacked General 4's battalion on planet 0. 8 troops were lost.

Median Output

If and only if the `--median/-m` option is specified on the command line, at the times detailed in the Galaxy Logic section, your program should print the current median troops lost in a battle for all planets in ascending order by planet ID. **If no battles have occurred on a specific planet, do not print the median message for that planet.** In the case that battles have occurred, you should print:

Median troops lost on planet <PLANET_ID> at time <TIMESTAMP> is <MED_TROOPS_LOST>.

If an even number of battles occur on a planet, take the average of the middlemost two to compute the median. There will always be an even number of troops lost in a battle (the same number of troops from both sides), so the result will always be an integer, even when divided by two.

Example

Given the following battles:

General 5's battalion attacked General 4's battalion on planet 9. 8 troops were lost.
General 2's battalion attacked General 7's battalion on planet 9. 2 troops were lost.

The median lost troops for planet 9 after these two battles is $((2 + 8) / 2) = 5$. If the timestamp changed, and the `--median` option was specified, your program should print:

Median troops lost on planet 9 at time 0 is 5.

Summary Output

After all input has been read and all possible battles have been fought, the following output should **always** be printed without any preceding newlines before any optional end of day output:

---End of Day---

Battles: <NUM_BATTLES><NEWLINE>

<NUM_BATTLES> is the total number of battles that have happened over the course of the program.

General Evaluation Output

If and only if the `--general-eval/-g` option is specified on the command line, following the summary output, you should print the following line without any preceding newlines.

---General Evaluation---

Followed by lines in the following format for **every** general in ascending order (0, 1, 2, etc.), even if they did not engage in any battles:

General <GENERAL_ID> deployed <NUM_JEDI> Jedi troops and <NUM_SITH> Sith troops,

and `<NUM_SURVIVORS>/<NUM_DEPLOYED>` troops survived.

These numbers are troops across all planets. Example:

---General Evaluation---

General 0 deployed 40 Jedi troops and 0 Sith troops, and 20/40 troops survived.

General 1 deployed 0 Jedi troops and 0 Sith troops, and 0/0 troops survived.

General 2 deployed 60 Jedi troops and 30 Sith troops, and 44/90 troops survived.

Movie-Watcher Output

As a fervent Star Wars fan, you want to find which pairs of battling Jedi and Sith deployments would have made for a maximally-exciting movie. For each planet, your job is to find the pairs of Jedi and Sith deployments that would have had the most “exciting” battles. The most “exciting” battle is one that has the greatest difference in Force-sensitivity between the battling parties. A battle can only happen if the Jedi Force-sensitivity is less than or equal to the Sith Force-sensitivity.

You want to find the most exciting possible Sith **attack** and Sith **ambush** for each planet. A Sith **attack** occurs when the Jedi are deployed to the planet first, and the Sith are deployed to that planet afterward to attack the Jedi. A Sith **ambush** occurs when the Sith are deployed to the planet first, and they surprise Jedi that land on the planet afterward. One deployment comes “after” another one if it appears later in the file, even if the timestamps for the two deployments are the same.

For example, suppose you had these deployments:

```
0 JEDI G1 P0 F10 #10
0 SITH G2 P0 F20 #10
0 JEDI G1 P0 F30 #10
0 SITH G1 P0 F40 #10
```

Then the most exciting Sith attack on planet 0 would be the Sith with Force-sensitivity 40 attacking the Jedi with Force-sensitivity 10, but there would be no most-exciting Sith ambush because it's not possible for a the Sith deployment to be paired against a Jedi deployment that came later.

Notice that an actual battle need not happen. In this output, the first and second deployments would be paired in the regular output, not the first and the fourth. You should report only the most exciting hypothetical battle, regardless of which battles actually occurred.

If and only if the `--watcher/-w` option is specified on the command line, you should print the following line without any preceding newlines once at the very end of the program :

---Movie Watcher---

Followed by a pair of movie watcher's output lines for every planet in ascending order in the following format:

A movie watcher would enjoy an ambush on planet <PLANET_ID> with Sith at time <TIMESTAMP1> and Jedi at time <TIMESTAMP2> with a force difference of <NUM>. A movie watcher would enjoy an attack on planet <PLANET_ID> with Jedi at time <TIMESTAMP1> and Sith at time <TIMESTAMP2> with a force difference of <NUM>.

When finding exciting battles, the number of troops is not taken into consideration. If there would be more than one battle with the maximal level of excitement (i.e., difference in Force-sensitivity), you should prefer the battle with the lower <TIMESTAMP2>. If the second timestamps are equal, then use the lower <TIMESTAMP1>.

If there are no exciting battles (there are no pairs of Jedi/Sith deployments on a given planet, or none result in the Jedi Force-sensitivity being less than or equal to the Sith Force-sensitivity) you should print -1 for both the first and second timestamp.

Detailed Algorithm

Following these steps in order will help guarantee that your program prints the correct output at the proper times.

The CURRENT_TIMESTAMP starts at 0, and is maintained throughout the run of the program.

1. Print program startup output
2. Read the next deployment from input.
3. If the new deployment's TIMESTAMP is not the CURRENT_TIMESTAMP
 - a. If the --median option is specified, print the median information
 - b. Set CURRENT_TIMESTAMP to be the new deployment's TIMESTAMP.
4. Instigate all possible fights between the Jedi and Sith battalions.
 - a. If the --verbose option is specified, you should print the details of each completed battle to stdout/cout.
5. Repeat steps 2-4 until there are no more deployments to be made.
6. Output median information **again** if the --median flag is set.
7. Print end-of-day summary output.
8. Output the general evaluation if the --general-eval flag is set.
9. Output the movie-watcher's output if the --watcher flag is set.

Full Example

Input File Contents:

```
COMMENT: DL mode generating some deployments.
MODE: DL
NUM_GENERALS: 3
NUM_PLANETS: 2
0 JEDI G0 P1 F100 #44
1 JEDI G0 P1 F56 #42
```

```
2 SITH G0 P1 F73 #19
2 SITH G0 P0 F34 #50
2 JEDI G0 P0 F86 #23
2 JEDI G0 P0 F20 #39
2 SITH G2 P0 F49 #24
2 JEDI G2 P0 F83 #45
3 JEDI G2 P1 F64 #22
3 JEDI G1 P0 F6 #19
3 JEDI G0 P1 F42 #37
4 JEDI G2 P0 F10 #44
```

Output when run with -v, -m, -g, and -w:

Deploying troops...

General 0's battalion attacked General 0's battalion on planet 1. 38 troops were lost.

General 0's battalion attacked General 0's battalion on planet 0. 78 troops were lost.

Median troops lost on planet 0 at time 2 is 78.

Median troops lost on planet 1 at time 2 is 38.

General 2's battalion attacked General 1's battalion on planet 0. 38 troops were lost.

Median troops lost on planet 0 at time 3 is 58.

Median troops lost on planet 1 at time 3 is 38.

General 2's battalion attacked General 2's battalion on planet 0. 10 troops were lost.

General 0's battalion attacked General 2's battalion on planet 0. 22 troops were lost.

Median troops lost on planet 0 at time 4 is 30.

Median troops lost on planet 1 at time 4 is 38.

---End of Day---

Battles: 5

---General Evaluation---

General 0 deployed 185 Jedi troops and 69 Sith troops, and 127/254 troops survived.

General 1 deployed 19 Jedi troops and 0 Sith troops, and 0/19 troops survived.

General 2 deployed 111 Jedi troops and 24 Sith troops, and 95/135 troops survived.

---Movie Watcher---

A movie watcher would enjoy an ambush on planet 0 with Sith at time 2 and Jedi at time 3 with a force difference of 43.

A movie watcher would enjoy an attack on planet 0 with Jedi at time 2 and Sith at time 2 with a force difference of 29.

A movie watcher would enjoy an ambush on planet 1 with Sith at time 2 and Jedi at time 3 with a force difference of 31.

A movie watcher would enjoy an attack on planet 1 with Jedi at time 1 and Sith at time 2 with a force difference of 17.

Hints and Advice

The project is specified so that the various pieces of output are all independent. We strongly recommend working on them separately, implementing one command-line option at a time. The autograder has some test cases which are named so that you can get a sense of where your bugs might be.

We place a strong emphasis on time budgets in this project. This means that you may find that you need to rewrite sections of your code that are performing too slowly or consider using different data structures. Pay attention to the Big-O complexities of your implementation and examine the tradeoffs of using different possible solutions. Using `perf` on this project will be incredibly helpful in finding which parts of your code are taking up the most amount of time, and remember that `perf` is most effective when your code is well modularized (i.e. broken up into functions).

Running your code locally in `valgrind` can help you find and remove undefined (buggy) behavior and memory leaks from your code. This can save you from losing points in the final run when you mistakenly believe your code to be correct.

It is extremely helpful to compile your code with the following gcc options: `-Wall -Wextra -Werror -Wconversion -pedantic`. This way the compiler can warn you about parts of your code that may result in unintended/undefined behavior. Compiling with the provided Makefile does this for you.