HOMEWORK #7:

Robots Party

Due Date: Wednesday, April the 6th, 11:59:59pm

For this assignment, submit any necessary files, but your '`main()`' function should be in a file called '`robotparty.cpp`'. Your program will be **compiled** and tested using the command '`fg++ *.cpp`', so please don't submit extra files that may interfere with compilation. Remember to put your **name** and **section** at the top of your program files. Your program should expect all input to come from '`cin`', and all your output should be to '`cout`'.

Problem:

Every morning, robots line up at the entrance of *"Mom's Robot Service & Maintenance"* shop to enjoy some of the services offered. The shop offers 4 different services: **Metalworking**, to remove dents and dings; **Painting**, for a shiny new look; **Memory wiping**, for those who know too much, and **Oil Bathing**, a decadent treat. Robots receive services in the order they are lined up. A robot may want to use one or more of these services. However, only **one** robot can receive a particular service at the same time, and after receiving a service, if a robot has other needs, it must go to the **back** of the line.

Your job is to write a program that simulates the robots coming in and out of the shop, and predict when each robot receives a service, when do they leave the shop, and how much time it takes the shop to service all robots.

Input:

The input will consist of a list of the robots lined up at the entrance of the shop. Each robot has a name, a number of services it wants, and the list of services it wants. The end of the list of robots lined up is signaled with a '`$`' character.

Output:

Consider a clock that starts at 0 when the shop opens. Output the time at which each robot receives a service, moves back to the back of the queue, or leaves the shop.

Simulation Details:

- The services will be denoted by the strings **"**`paint`**"**, **"**`metalwork`**"**, **"**`oilbath`**"** and **"**`memwipe`**"**.
- Painting a robot takes **18** minutes.

- Metalwork takes **12** minutes to finish a robot.
- An oil bath takes **9** minutes to completion.
- A Memory Wipes takes just **4** minutes to flash a robot's mind squeaky clean.
- A robot who gets a Memory Wipe **forgets** the other services it wanted and leaves the shop.
- Robot names **do not** contain spaces.

Sample:

Input

```
Bender   1 paint
Calculon 4 metalwork memwipe metalwork paint
r2d2     2 oilbath paint
c3p0     3 memwipe oilbath paint
Hedonismbot 5 oilbath oilbath oilbath paint oilbath
TinyTin  2 metalwork metalwork
$
```

Output

```
At time 0 : Bender gets: paint.
At time 0 : Calculon gets: metalwork.
At time 0 : r2d2 gets: oilbath.
At time 0 : c3p0 gets: memwipe.
At time 4 : r2d2 gets back in line for: paint.
At time 4 : Hedonismbot gets: oilbath.
At time 8 : Hedonismbot gets back in line for: oilbath.
At time 9 : c3p0 is done!
At time 12 : Calculon gets back in line for: memwipe.
At time 12 : TinyTin gets: metalwork.
At time 18 : Bender is done!
At time 18 : r2d2 gets: paint.
At time 18 : Hedonismbot gets: oilbath.
At time 18 : Calculon gets: memwipe.
At time 22 : Hedonismbot gets back in line for: oilbath.
At time 22 : Hedonismbot gets: oilbath.
At time 24 : TinyTin gets back in line for: metalwork.
At time 24 : TinyTin gets: metalwork.
At time 26 : Hedonismbot gets back in line for: paint.
At time 27 : Calculon is done!
At time 36 : r2d2 is done!
At time 36 : Hedonismbot gets: paint.
At time 36 : TinyTin is done!
At time 54 : Hedonismbot gets back in line for: oilbath.
At time 54 : Hedonismbot gets: oilbath.
```

```
At time 58 : Hedonismbot is done!
```

Implementation Guidelines:

- Your queue implementation should be a subclass of the provided "`AbstractQueue`" class.
- Implement and **test** your queue **before** you code the simulation.
- Think about how you are going to represent Robots and Services. A class '`Robot`' and a class '`ServiceStation`' may prove useful.
- When you grant a robot a service, keep track of when than service will be completed.
- **Understand** the sample input/output **before** you design your program.
- This will prove to be a **challenging** homework. Start early!!
- A suggestion for your simulation algorithm is provided in the next section:
- Think about how you handle the exception (throw/try/catch, if needed)
- Use a '*.h' and '*.hpp' file pair for each of your classes.
- Compress files to submit together

Simulation Algorithm Draft:

```
WHILE the queue is not empty OR the services are busy.

    IF the queue is not empty
        Let 'r' be the robot at the front of the queue.
        Let 'm' be the service requested by 'r'

    IF the queue is not empty AND 'm' is available
        Give 'r' service 'm'
        Record time of completion of service 'm'
    ELSE
        Let 'w' be the service that will finish earliest.
        Let 'x' be the robot receiving 'w'
        Advance the clock to the time 'w' finishes
        IF 'x' wants more services
            place 'x' at the back of the queue
        ELSE
            'x' is done!
```