

**Important:** The grader will compile and run your lab only on the stdsun (so even if you develop it on a different computer, it is your responsibility, before submitting, to make sure that it runs as you intend it to, on the stdsun).

**Grade:** This lab is worth 25 points.

**Lab:** In this lab, you will modify the `Robot` class from the previous lab. You will create a *base* `Robot` class and three derived classes, `NormalRobot`, `LazyRobot`, and `EagerRobot`. Instances of the `NormalRobot` class will behave like instances of the `Robot` class of Lab 2. Instances of the `LazyRobot` class are *lazy*; when asked to move  $n$  steps (by using the `move()` function), they move only  $(n - 1)$  steps (0 if  $n$  is 1). Instances of the `EagerRobot` class are *eager*; when asked to move  $n$  steps, they move  $(n + 1)$  steps. Hence the `move()` function should be defined as a *pure virtual* function in the base class with the actual definitions corresponding to each kind of robot being in the corresponding derived class.

The constructor functions in the derived classes should invoke the corresponding constructors of the base class. In addition, each should output to `cout`, a message such as:

```
New lazy robot created at coordinate position: ..; direction: ...
```

The other functions should be as in Lab 2.

The `main()` function, as in Lab 2, should define an array `myRobots` of 20 elements each of which is a *pointer* to a `Robot`. As before, these elements should all be initialized to 0 (i.e., the null pointer). As before, the `main()` function should read in a series of lines from the standard input and take appropriate action. The key difference is that whereas in the previous lab, if the first number of the input line was 1, that indicated that a new robot was to be created, now creation will be indicated by the first number being 11, 12, or 13; if the number is 11, that means create a new `normalRobot`; if it is 12, that means create a new `lazyRobot`; and if it is 13, that means create a new `eagerRobot`.

If the input line is `2 k m 0`, as before that indicates that robot  $k$  must be moved by  $m$  steps; or rather, the `move()` operation must be invoked on this robot with  $m$  as the argument. How many steps the robot will *actually* move will, of course, depend on whether this is a `normalRobot`, a `lazyRobot`, or an `eagerRobot`.

One other difference is when the input line is `6 0 0 0`. As before, in this case you should output the coordinates and direction of each robot but also, for each robot, you should include what kind of robot it is. This should be done by introducing an appropriate function in the base class and defining it appropriately in the derived classes, *not* by keeping track, in the `main()` function, of the kind of robot that each element of the `myRobots` array points to.

**What To Submit And When:** On or before 11:59 pm, March 12, you should submit the following: a single file named `lab3.cpp` that contains the definitions of all your functions, including the `main()` function. If you decided to define additional functions –and it is certainly appropriate to do so if that helps improve the structure and understandability of your code– the definitions of those functions should also be included in the same file. You should not have to use any library other than `<iostream.h>` If you did use another library, explain that in the documentation. Overall documentation for the lab should appear as a set of C++ comments at the start of the `lab3.cpp` file. Additional documentation should appear as a brief explanation at the start of the `Robot` class and at the start of each function to explain what it does and how it works.

Submit your lab using the following submit command (assuming `lab3.cpp` is the name of the file you want to submit):

```
submit c459.22ac lab3 lab3.cpp
```

If you have questions or comments or notice problems in the above, please email me.