**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 write a Robot class. Instances of this class will represent simple robots on a two-dimensional grid. A robot has position which consists of x- and y-coordinates (both non-negative integers) and a direction (one of the values East, North, West, or South). The Robot class should provide the following public member functions: void move(int n) which will move the robot, in the direction it is currently facing, by n steps; void turnLeft() which will change the robot’s direction appropriately (for e.g., if the current direction is North, after this operation, it will be West); void turnRight() which is similar and turns the robot to the right; int xCoor() which returns the current x-coordinate; int yCoor() which returns the current y-coordinate; char direction() which returns the current direction (E, N, W, or S). The three member functions that return information about the robot’s position be declared as const. The class should have two constructor functions; the first is the default constructor, should accept no parameters and should initialize the new robot to be at the origin (i.e., x- and y-coordinates zero) and facing North; the second should receive three parameters, the first two being integers (the initial values of x- and y-coordinates) and the third being of type char whose value should be one of E, N, W, S, specifying the initial direction.

The main() function should define an array myRobots of 20 elements each of which is a pointer to a robot. These elements should all be initialized to 0 (i.e., the null pointer). The main task of the main() function should be to read in a series of lines from the standard input and take appropriate action as follows. If the line is of the form:

- **1 x y d:** This means create a new robot at the coordinates \((x, y)\) and facing direction \(d\) where the value of \(d\) must be one of 1, 2, 3, or 4 denoting East, North, West, South. If the given value of \(x\) or \(y\) is negative or that of \(d\) is not one of \(\{1, 2, 3, 4\}\), output an appropriate error message and also state that the robot will be created at the default position/direction. If 20 robots are already in existence, output an error message (saying “no room for more robots”) and continue to the next line. Else create a new robot with the right (or default) coordinates and direction, store a pointer to it in one of the currently unused element \(k\) of myRobots and output a message saying, “robot created; robot number: ...”.

- **2 k m 0:** This means move the robot number \(k\) by \(m\) steps in the direction it is currently pointing. If the value of \(k\) is below 0 or above 19, output an error message (saying “no such robot”) and continue to the next line; similarly if myRobots currently has no robot in its \(k\)th element. Otherwise move this robot \(m\) steps in the direction it is pointing; but neither the \(x\) nor \(y\) coordinate is allowed to be negative so if, following the move, either coordinate would become negative, instead of moving, delete the robot and print a message saying, “robot no.:... walked off the board!”. Otherwise, print a message saying, “robot no.:... moved; new coordinates and direction: ...”.
• 3 k d 0: This means turn the robot number k so it is facing East, North, West, or South if d is 1, 2, 3, or 4 respectively. If d is not one of these values, print an appropriate message and continue to the next line; otherwise invoke the necessary function(s) to turn the robot appropriately and print a message saying, “robot no.:... turned; new coordinates and direction: ...”.

• 4 k d 0: This means display the position of robot number k; if there is no robot in the kth element of myRobots or k is outside the proper range, output a message saying, “no such robot”. Else print a message saying, “the coor. and direction of robot no....: ...”

• 5 k 0 0: This means kill robot number k. If there is no such robot, output an appropriate message; if not, delete the robot and output a message to that effect.

• 6 0 0 0: This means output the number of robots currently in existence and output each one’s coordinates and direction.

• 7 0 0 0: This means “game over”. Output a suitable message and terminate the program ... but remember to delete all the currently existing robots; else you will have a memory leak.

What To Submit And When: On or before 11:59 pm, March 2, you should submit the following: a single file named lab2.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 lab2.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 lab2.cpp is the name of the file you want to submit):

```
submit c459.22ac lab2 lab2.cpp
```

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