overview

- This is the project for unit II of cis 1.5. This project covers reading simple information and making decisions about it (i.e., input and control structures).
- The project is worth 10% of your term grade. It will be marked out of **10 points**.
- The project is due via email on **Monday March 16**.
- Email the project to: `sklar@sci.brooklyn.cuny.edu`.
- Use a **zip** utility to bundle all your files together and send them as ONE attachment to the email.
  on a PC: use **WinZip**
  on a Mac: use **File - Create Archive...**
  on Linux: use **zip**
- Note that if you are using an IDE, all your C++ source code files will be named **main.cpp**. You will have to rename them before putting them in the zip archive!

project description

This project is based on the **roomba.cpp** program that you have been working on for labs II.1 through II.3. Similarly to the labs, you will be writing a program that simulates a scribbler robot moving around a room. The room is 12 units across (in the x direction, or east/west) and 9 units down (in the y direction, or north/south). Assume that $(0, 0)$ is the upper left (farthest northwest) corner of the room, and $(11, 8)$ is the lower right (farthest southeast) corner of the room.

The project contains 5 parts. Each part is worth 2 points.

1 getting started

1.1. Create a new C++ program file called **scribbler.cpp**.

1.2. At the top of the file, be sure to include the following *header* files:

```
#include <stdlib.h>
#include <sys/time.h>
#include <iostream>
using namespace std;
```

1.3. Make yourself a template by puting in the code for the **main()** function:

```
int main() {
.
.
.
} // end of main()
```

1.4. Inside the **main()** function, at the beginning, declare the following variables:

- two integers that store the scribbler robot's x and y **home** location in the room; the "home" location is where the robot starts when the program begins;

- two integers that store the scribbler robot's x and y **current** location in the room; the current location is wherever the robot moves to each time the user enters a valid command;
- a character variable to store the user's input; and
- a boolean variable that indicates if the user wants to exit the program or not.

2 finding home

2.1. Inside the **main()**, initialize the random number generator using the **srand()** function. Use the current time as the seed (using the function call **time(NULL)**).

2.2. Initialize the scribbler robot's **home** x and y location using calls to the **rand()** function.

2.3. Remember that the room is only 12×9 units in size, so x must be between 0 and 11 and y must be between 0 and 8. (Refer to the lecture notes from Feb 25 for an example of how to use the modulo $(\%)$ operator to scale the output from **rand()**.)

2.4. Initialize the scribbler robot's **current** x and y location to be the same as its home location (set above).

3 reading the user's input

3.1. Inside the **main()**, create a **while** loop, using the boolean variable (declared above) to control the loop. Remember to initialize the boolean variable *outside* the loop, before the **while** statement.

3.2. Inside the **while** loop, ask the user to enter any of five commands: N to make the robot go north, S to make the robot go south, E to make the robot go east, W to make the robot go west, or X to exit the loop.

3.3. Read the user's input and *echo* it back to the screen (e.g., `you just entered N`).

4 acting based on the user's input

4.1. Inside the **while** loop, branch according to what the user entered, as follows:

- if the user entered N or n, make the robot move "north" by decreasing the robot's y value by 1;
- if the user entered S or s, make the robot move "south" by increasing the robot's y value by 1;
- if the user entered E or e, make the robot move "east" by increasing the robot's x value by 1;
- if the user entered W or w, make the robot move "west" by decreasing the robot's x value by 1;
- if the user entered X or x, set the boolean loop control variable to exit the **while** loop;
- otherwise (the user entered none of the above values), display a message telling the user that s/he entered an invalid value and let her/him try again.

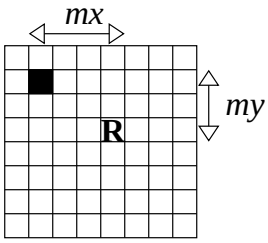
4.2. You can use either an **if/else** or a **switch/case** control structure to handle the branching.

4.3. When you increment or decrement the robot's x and y values, make sure to check that the robot does not go outside the boundaries of the room. *Do NOT use wrap-around for this assignment.*

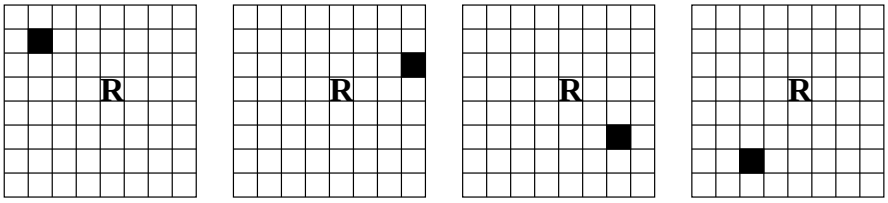
4.4. If the user enters a valid command other than X or x, display the robot's current location.

5 calculating the distance home

5.1. The Manhattan distance is the total number of units in the x direction and in the y direction that the robot must travel to go from one location to another. Since the robot cannot move diagonally (it can only move north, south, east or west), it has to travel as if it were walking the square blocks in Manhattan. An example is below. If the robot's home location is (hx, hy) and the robot's current location is (x, y), then the value of $mx = x - hx$ and the value of $my = y - hy$. The Manhattan distance $= mx + my$.



5.2. Be aware that there are a number of possibilities for how the home location and the robot's current locations can be related to each other in the room.
Consider the drawings below:



(a) (b) (c) (d)

In these drawings, the robot is represented by the letter **R** in the grid and the charging station (home) is represented by a black square. In figure (a), the home is to the northwest of the robot. In figure (b), the home is to the northeast of the robot. And so on.

5.3. The Manhattan distance should always be a *positive* number. So in case (a) above, $mx = x - hx$ and $my = y - hy$, as shown earlier. But in case (b), if we calculated $mx = x - hx$, we would get a negative number, since $hx > x$. In this case, you need to calculate $mx = hx - x$.
→ For each of the four cases above, write down the formulas for calculating mx and my.

5.4. Next you need to know which case you have.
→ For each of the four cases above, write down the relationships between x and hx and y and hy that go with each case. For example, for case (a), $x > hx$ and $y > hy$.

5.5. Now, you are ready to put this in your code. If the user enters a valid command other than X or x, calculate the *Manhattan* distance to the robot's **home** location from its **current** location. First determine which case you have. Then use the formulas you determined above that go with your case. Then display the Manhattan distance. Be sure to include a friendly message explaining what you are displaying, like: `the Manhattan distance to the robot's home is: 5 units`.

submission instructions

- You will be submitting ONE file: **scribbler.cpp**
- Make sure that you have a COMMENT at the top of the file that contains the name of the file, YOUR NAME, "CIS 1.5 PROJECT 2" and the submission date (March 16, 2009).
- The subject line of your email should say: CIS 1.5 PROJECT 2