Assignment #4 –
Templates, Inheritance, Vectors, and Exceptions
Due: Tuesday, 05/26/15, 11:59pm

You are the owner of a new business idea called the Roomba!!! You are trying to determine the best/most efficient algorithm to program for your Roomba operation. You want the Roomba to not only clean the entire floor, but in the fastest time possible. Your Roomba can only go left, right, up, and down from its current location. It has a battery life that we will simulate as a maximum number of moves. In addition, the Roomba can be placed on any spot on the floor.

Your program will take the name of a file, battery life, and starting position as command line arguments with the –f (filename), -b (battery), -v (moves before viewing), –r (starting row) and –c (starting column) options.

```
a.out –f floorplan.txt –b 100 –r 0 –c 0 –v 1
```

Catch Exceptions with File I/O:
You will try to open the file for input, but if it is not present, then you must catch the exception and let the user know that he/she must supply a floor plan file in order to run the simulation. The file with the floor plan doesn't contain dimensions. It only contains the cells of floor and potential objects, indicated by 'f' or 'd' for floor (or dirty floor) and 'o' for object. For example:

```
f f f o
f f f o
o o f f
f f f f
```

For this Assignment, your floor doesn't need to have objects, it can just be a dirty floor that needs to be cleaned!

Use classes with a Template, "Is a", "Has a", and "Uses" relationship:
Here are the classes you need to have:

```
template <typename T>
class Room {
   private:
      Floor f;
      T *obj; //obj could be a Roomba, Person, etc.
   public:
      //The functionality of a room is to view it, put
      //objects in it, take objects out of it, etc.
      …
      void clean_room() {
         if(obj!=NULL)
              obj->clean_floor(f);
      }
};
```

```
class Floor {
   private:
      vector < vector <char> > v;
      int obj_row; //Holds an object's current row position
      int obj_col; //Holds an object's current col position
   public:
      //The functionality of a floor is to create the floor
      //plan, as well as determining if an object's
      //position is at the edge of the room
       …
};
class Roomba {
   private:
      int battery;      //how many moves it can make
      int total_moves;  //total moves needed to clean
   public:
      //The Roomba needs to be able to clean a floor, and
      //move left, right, up, down
      …
};
class RandomRoomba : public Roomba {
   private:
      …
   public:
      …
};
class …Roomba : public Roomba {  //for first AI Roomba
class …Roomba : public Roomba {  //for second AI Roomba
```

You can decide the function names for each class and whether you need to define the Big Three for the class. However, you MUST have the class relationship, members, and functionality specified above. Be creative!!!

Simulate the Roomba:

Now, you need to simulate the Roomba cleaning the room. You know you have a Room with a Roomba that cleans a specific floor. It is your job in your simulation to find which Roomba can clean the room the fastest (or in the least amount of moves). Here is the simulation you need:

```
#include <iostream>
#include "./Room.hpp"
#include "./RandomRoomba.h"
…
using namespace std;

void choose_roomba(Room <Roomba> &r) {
```

```
   int choice;
   cout << "Which Roomba (1 – Random, 2 - ?, 3 - ?)? ";
   cin >> choice;

   if(choice == 1)
      r.set_roomba(new RandomRoomba);
   else if(choice ==2)
   …
}
void simulate_roomba() {
   Room <Roomba> r;

   choose_roomba(r);
   //More setup is needed for the Room
   …
   r.clean_room(); //Now clean it after everything is setup
   …
}
int main(int argc, char *argv[]) {
   simulate_roomba();

   return 0;
}
```

Example output:

After running a program with the file and command line arguments provided below, your program simulation might look something like the output below. (This is a very smart random!)

```
a.out –f floorplan.txt –b 100 –r 0 –c 0 –v 1

Which Roomba (1 – Random, 2 - ?, 3 - ?)? 1
Beginning:
r f f f
f f f f
f f f f

Move 1:
c r f f
f f f f
f f f f

Move 2:
c c r f
f f f f
f f f f
```

Move 3:
c c c r
f f f f
f f f f

Move 4:
c c c c
f f f r
f f f f

Move 5:
c c c c
f f f c
f f f r

Move 6:
c c c c
f f f c
f f r c

Move 7:
c c c c
f f r c
f f c c

Move 8:
c c c c
f r c c
f f c c

Move 9:
c c c c
r c c c
f f c c

Move 10:
c c c c
c r c c
f f c c

Move 11:
c c c c
c c c c
f r c c

```
Move 12:
c c c c
c c c c
r c c c

Entire floor visited and cleaned in 12 moves!!!
```

Things to consider:

- You must keep the classes, members, and functionality specified in the design above.
- Your member variables in all classes must be private or protected for encapsulation rules.
- You must have your class definitions in a .h file and your implemented classes in .cpp files. Your template class can have interface and implementation in a .hpp file.
- You must also have your main function in a simulate_roomba.cpp file, separated from the class implementations.
- Create a **Makefile** for you project.

(10 pts) Extra Credit to avoid objects:

After running a program with the file and command line arguments provided above, your program simulation might look something like the output below. (This is the best random ever!)

```
Which Roomba (1 – Random, 2 - ?, 3 - ?)? 1
Beginning:
r f f o
f f f o
o o f f
f f f f

Move 1:
c r f o
f f f o
o o f f
f f f f

Move 2:
c c r o
f f f o
o o f f
f f f f


Move 3:
c c c o
f f r o
o o f f
f f f f
```

Move 4:
c c c o
f r c o
o o f f
f f f f

Move 5:
c c c o
r c c o
o o f f
f f f f

Move 6:
c c c o
c r c o
o o f f
f f f f

Move 7:
c c c o
c c r o
o o f f
f f f f

Move 8:
c c c o
c c c o
o o r f
f f f f

Move 9:
c c c o
c c c o
o o c r
f f f f

Move 10:
c c c o
c c c o
o o c c
f f f r

```
Move 11:
c c c o
c c c o
o o c c
f f r c

Move 12:
c c c o
c c c o
o o c c
f r c c

Move 13:
c c c o
c c c o
o o c c
r c c c

Entire floor visited and cleaned in 13 moves!!!
```

(10 pts) **Program Style/Comments**

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!

http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/162_style_guideline.pdf

```
/*************************************************
** Program: simulate_roomba.cpp
** Author: Your Name
** Date: 05/20/2015
** Description:
** Input:
** Output:
*************************************************/
```

(10 pts) Design for Assignment #4 changes/Testing

- **(5 pts) How did your design for the Roomba choices in Assignment #4 change during implementation?**

- **(5 pts) What were the actual values from your testing? Did these match your expected values? What did you do to make sure you get the expected values?**

  Please see the template for this document: Polya_template.pdf

  You need to have a table with these headings:

Input Values	Expected Output	Did Actual Meet Expected?
t	Error message for choice ‘t’ because that isn’t an option. Reprompt for choice again.	Yes

In order to submit the files, you will be creating a bzipped tar ball. In order to do this, you will use the following command, adding all the source files to the end of the command:

```
tar -cjvf cs162_hwx_username.tar.bz2 file1 file2 file3...
```

This tar ball (replacing username with your ENGR username), and only this tar ball, will be submitted via TEACH.