Project 3, Hare and Tortoise Competition, Spring 2024

1 Requests

Write a hare and tortoise competition program using object-oriented programming. Keys to learn for this project are class and object design, encapsulation, and vector.

1. There is a road with n blocks, where n >= 1. The road can be slippery, so occasionally animals may move backwards from current position.

2. The first animal to reach or pass the last block is the winner; there can also be a tie.

3. In each round, your program adjusts the position of the animals according to its moving pattern. Use variables to keep track of animals' current positions, which can be at block 0 to n-1.

4. Each animal starts at position 0 (the first block). If an animal slips left before block 0, move that animal back to block 0. You may image there is a road blocker cone at the beginning of the first block, hence animals cannot fall left to block 0.

5. Similarly, if an animal moves past position n-1 (the last block), move that animal back to square n-1. You may image there is a road blocker cone at the end of last block, hence animals cannot fall right to last block.

6. Here are how the hare and tortoise can move typically.

Animal	Move Type	Percentage of the time	Actual move
Tortoise	Fast plod	50%	3 squares to the right
	Slip	20%	6 squares to the left
	Slow plod	30%	1 square to the right
Hare	Sleep	20%	No move at all
	Big hop	20%	9 squares to the right
	Big slip	10%	12 squares to the left
	Small hop	30%	1 square to the right
	Small slip	20%	2 squares to the left

7. Generate the percentages in the above table by producing a random integer in the range $0 <= i < 10$. For the tortoise, perform a "fast plod" when $0 <= i < 5$, a "slip" when $5 <= i < 7$ or a "slow plod" when $7 <= i < 10$. Use a similar technique to move the hare.

8. Begin the race.

(a) For each round in competition, display a line whose length is the same as number of blocks in a road. Letter T is shown in the position of the tortoise and the letter H in the position of the hare.

(b) Occasionally, the animals can land on the same block. In this case, the tortoise bites the hare, and your program should display only letter T, with message "Ouch, Tortoise bites Hare." in the next line.

(c) All output positions other than T and H should be blank.

9. After each line is displayed, test for whether either animal has reached or passed the last block. If so, display the winner and terminate the simulation.

   (a) If the tortoise wins, display
   Yay!!! tortoise wins!

   (b) If the hare wins, display
   Yuck, hare wins.

   (c) If both animal win on the same tick of the clock, you can print out
   It is a tie.

10. Write comments, which include but not limit to the author's name, email, purpose, key ideas and sample output.

11. You need to finish this project by yourself.

2 Task A

`Hare.hpp` is provided.

```cpp
#ifndef Hare_H
#define Hare_H
#include <vector>

class Hare {
public:
   Hare();
      //a typical hare that sleeps 20% of the time,
      //move 9 squares to the right 20% of the time,
      //move 12 squares to the left 10% of the time,
      //move 1 square to the right 30% of the time,
      //move 2 squares to the left 20% of the time.
   Hare(std::vector<int> patterns, int position);
   Hare(int* arr, int size, int position);
   void move();
   int getPosition() const;
   void setPosition(int position);
private:
    std::vector<int> patterns;
    int position;
};
#endif
```

Define `Hare.cpp`, which implements constructors and methods in `Hare.hpp`. Submit only `Hare.cpp` (no need to add main function) to gradescope.

2.1 Test Hare.cpp in local computers

1. Comment `private:` in Hare.hpp for data members.
2. Edit the following `main_test_hare.cpp`.

```cpp
#include <iostream>
#include <string>
#include "Hare.hpp"
//g++ -std=c++11 main.cpp would be fine with #include "Hare.cpp"
//#include <vector> //included in "Hare.hpp" already
using namespace std;

int main(int argc, const char *argv[]) { //does not take input from users
    Hare hare;

    vector<int> patterns = {1, -2, 6}; //c++11
    Hare hare2(patterns, 2);

    int arr[] = {1, 1, 2, -3, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    Hare hare3(arr, size, 6);

    switch (*argv[1]) {
    case 'A': //default constructor
         cout << "After calling default constructor, data member patterns is {";
         for (int i = 0; i < hare.patterns.size(); i++) {
             cout << hare.patterns[i];
             if (i < hare.patterns.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         cout << "After calling default constructor, data member position is ";
         cout << hare.position << endl;
         break;
         //a typical hare that sleeps 20% of the time,
         //move 9 squares to the right 20% of the time,
         //move 12 squares to the left 10% of the time,
         //move 1 square to the right 30% of the time,
         //move 2 squares to the left 20% of the time.
      case 'B': //Hare(std::vector<int> patterns, int position);
         cout << "After calling constructor Hare(std::vector<int>, int), data
            member patterns is {";
         for (int i = 0; i < hare2.patterns.size(); i++) {
             cout << hare2.patterns[i];
             if (i < hare2.patterns.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         cout << "After calling constructor Hare(std::vector<int>, int), data
            member position is ";
         cout << hare2.position << endl;
         break;
    case 'C': //Hare(int* arr, int size, int position);
         cout << "After calling constructor Hare(int*, int, int), data member
            patterns is {";
         for (int i = 0; i < hare3.patterns.size(); i++) {
             cout << hare3.patterns[i];
             if (i < hare3.patterns.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         cout << "After calling constructor Hare(int*, int, int), data member
            position is ";
         cout << hare3.position << endl;
         break;
    case 'D': //void move();
         cout << "After calling move method, data member position is ";
         hare.move();
         cout << hare.position << endl;
         break;
    case 'E': //int getPosition();
         cout << "After setting hare's position to be 10 and calling getPosition
            method, data member position is ";
         hare.position = 10;
         cout << hare.getPosition() << endl;
         break;
    case 'F': //void setPosition(int position);
         cout << "After hare.setPosition(3); data member position is ";
         hare.setPosition(3);
         cout << hare.position << endl;
         break;
    default:
         cout << "wrong option";
    }

    return 0;
}
```

3. Compile and run the code.

```
g++ -std=c++11 main_test_hare.cpp Hare.cpp
```

If the above command runs successfully, run

(a) Test default constructor.
Run command
`./a.out 'A'`
You should see

```
After calling default constructor, data member patterns is
   {0,0,9,9,-12,1,1,1,-2,-2}
After calling default constructor, data member position is 0
```

(b) Test non-default constructor `Hare(std::vector<int>, int)`.
Run command
`./a.out 'B'`
You should see

```
After calling constructor Hare(std::vector<int>, int), data member patterns is
   {1,-2,6}
After calling constructor Hare(std::vector<int>, int), data member position is
   2
```

(c) Test non-default constructor `Hare(int*, int, int)`.
Run command
`./a.out 'C'`
You should see

```
After calling constructor Hare(int*, int, int), data member patterns is
   {1,1,2,-3,5}
After calling constructor Hare(int*, int, int), data member position is 6
```

(d) Test move method of Hare class.
Run command
`./a.out 'D'`
You should see similar result (result may vary).

```
After calling move method, data member position is 1
```

(e) Test `getPosition` method of Hare.

Run command

`./a.out 'E'`

You should see

```
After setting hare's position to be 10 and calling getPosition method, data
   member position is 10
```

(f) Test `setPosition` method of Hare.

```
After hare.setPosition(3); data member position is 3
```

3 Task B

`Tortoise.hpp` is provided.

```cpp
#ifndef Tortoise_H
#define Tortoise_H
#include<vector>
//normally we do not use namespace std; in hpp file
class Tortoise {
public:
    Tortoise();
    Tortoise(int* arr, int size, int position);
    Tortoise(std::vector<int> patterns, int position);
    void move();
    int getPosition() const;
    void setPosition(int position);

private:
    std::vector<int> patterns;
    int position;
};
#endif
```

Edit `Tortoise.cpp`, define constructors and methods declared in `Tortoise.hpp`. Submit `Tortoise.cpp` to gradescope. No main function is needed.

4 Task C

`Road.hpp` is provided.

```cpp
#ifndef Road_H
#define Road_H
#include <vector>
class Road {
```

```cpp
public:
    Road();
    Road(int length);
    void mark(int index, char ch);
    void display() const;
    int length() const;
private:
    std::vector<char> blocks;
};
#endif
```

Edit `Road.cpp`, define constructors and methods declared in `Road.hpp`. Submit `Road.cpp` to gradescope. No main function is needed.

- In the default constructor, instantiate data member `blocks` as a vector of 70 characters, where each element is a space character. You may utilize `assign` method of the vector class.
- In the non-default constructor, if the formal parameter **length** is non-positive, change it to be 70.

  Initialize data member `blocks` with **length** characters, with each element being a space character. You may use the `assign` method of the vector class.
- In the `mark` method, if the formal parameter `index` is a valid index in data member `blocks`, set (`index`) th elements of `blocks` to be the formal parameter `ch`.
- In the `display` method, print all elements of `blocks` in a line, followed by new line character.
- In the `length` method, return the number of elements in the data memer `blocks`.

4.1 Test Road.cpp in your computer

Take the following steps.

- Download `main_test_road.cpp`.

```cpp
#include <iostream>
#include <string>
#include "Road.hpp"
//g++ -std=c++11 main_test_road.cpp would be fine with #include "Road.cpp"
//#include <vector> //included in "Road.hpp" already
using namespace std;

int main(int argc, const char *argv[]) { //does not take input from users
    //Road();
    //Road(int length);

    Road rd;
    Road rd2(30);
    Road rd3(-1);

    switch (*argv[1]) {
    case 'A': //default constructor
         cout << "After calling default constructor, data member blocks is {";
         for (int i = 0; i < rd.blocks.size(); i++) {
             cout << "'" << rd.blocks[i] << "'";
             if (i < rd.blocks.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         break;
     case 'B': //Road(int)
         cout << "After Road(30), data member blocks is {";
         for (int i = 0; i < rd2.blocks.size(); i++) {
             cout << "'" << rd2.blocks[i] << "'";
             if (i < rd2.blocks.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         break;
    case 'C': //Road(int);
         cout << "After Road(-1), data member blocks is {";
         for (int i = 0; i < rd3.blocks.size(); i++) {
             cout << "'" << rd3.blocks[i] << "'";
             if (i < rd3.blocks.size()-1)
                cout << ",";
         }
         cout << "}" << endl;
         break;
    case 'D': //void mark(int index, char ch);
         cout << "After calling mark(2, 'H') method, blocks[2] is ";
         rd.mark(2, 'H');
         cout << rd.blocks[2] << endl;
         break;
    case 'E': //void display() const;
         cout << "After calling display() method with block indexed at 2 is marked
             by 'H' and block indexed at 5 is marked by 'T', data member blocks is "
             ;
         rd.blocks[2] = 'H';
         rd.blocks[5] = 'T';
         rd.display();
         break;
    case 'F': //int length() const;
         cout << "After calling length() method for Road rd2(30), data member
             length is ";
         cout << rd2.length() << endl;
```

```
            break;

        default:
            cout << "wrong option";
    }

    return 0;
}
```

- Download Road.hpp locally. Comment the line starts with private:. Need to uncomment this line after finishing testing Road.cpp.
- Define Road.cpp yourself, then run the following command:

  `g++ -std=c++11 Road.cpp main_test_road.cpp`

  If everything goes well, you would be able to get a.out file in Mac/Linux and a.exe in windows.
- Test default constructor of Road class, run the following command (note that there is a space between ./a.out and 'A')

  `./a.out 'A'`
- Test Road(int) constructor, run

  `./a.out 'B'`
- Test methods of Road class using a similar approach.

5 Task D

`Competition.hpp` is provided.

```
#ifndef Competition_H
#define Competition_H
#include "Hare.hpp"
#include "Tortoise.hpp"
#include "Road.hpp"

class Competition {
public:
    Competition();
    Competition(Hare coney, Tortoise cooter, int length);
    void play();

private:
    Hare rabbit;
    Tortoise tor;
    Road lane;
};
```

```
#endif
```

1. Edit `Competition.cpp`, define constructors and methods declared in `Competition.hpp`.
   (a) In the default constructor, data member `rabbit` is an object instantiated by default constructor of `Hare` class; data member `tor` is an object instantiated by default constructor of `Tortoise` class; and `lane` is an object instantiated by default constructor of `Road` class.
   (b) In the non-default constructor, data member `rabbit` is set to be the formal parameter **coney**, data member `tor` is set to be the formal parameter **cooter**, and data member `lane` is an object instantiated by non-default constructor of `Road` class that takes the formal parameter **length** in non-default constructor of `Competition` class as an actual parameter.
   (c) In method `play`, mimic the competition process.
      i. Find out the last block of the road.
      ii. Declare and initialize integer variable round to be 1.
      iii. Compete as long as neither `rabbit` nor `tor` reaches the last block of `lane`.
         A. `rabbit` moves. If, after move, rabbit's position is smaller than the index of the first block of `lane`, place `rabbit` to the first block of `lane`, otherwise, if rabbit's position is larger than the index of the last block of `lane`, place `rabbit` on the last block of `lane`. Hint: when we talk about moving `rabbit` to a specific block of `lane`, what method in Hare class should we use?
            When `rabbit` sets feet on a block, it leaves mark 'H' on the corresponding block of `lane`. Which method of `Road` class can be called?
         B. Move tortoise `tor`. Mark its position on the road by 'T'.
         C. Display round number and also the values of each element of `lane`.
         D. It may happen that `rabbit` and `tor` may step on the same block of `lane`. In that case, assume that `rabbit` is the first animal to move in each round, then `lane` can only keep the most recent mark left by the contestants, that is 'T'. In that case, print "Ouch. Tortoise bites hare." in a separated line.
         E. Prepare for the next round by cleaning the mark left by the current round. Which method of road class should be called?
         F. Increase round by 1.
      iv. Print competition result: tie, hare wins, or tortoise wins.
2. Submit `Competition.cpp` to gradescope.
3. No main function is needed.

6 Code of CompetitionTest.cpp

```
#include <iostream>
#include <string>
#include "Competition.hpp"
using namespace std;
```

```cpp

int main() {
    Competition race;
    race.play();

    vector<int> harePattern{-3, 2, 0, 0, 6}; //need c++11 or above
    Hare hare(harePattern, 0);

    vector<int> torPattern{1, 1, 2, -2, 3};
    Tortoise tor(torPattern, 5);

    Competition race2(hare, tor, 30);
    race2.play();
    return 0;
}
```

7 Run codes and submit

Important Change when submit

When you test your code in local computer, you can use `srand(time(null));` to randomize the running result when creating a Competition object. However, when you submit to gradescope, you need to remove `srand(time(null));` statement, Otherwise, gradescope cannot grade a random result.

1. Compile and run the project in local editors
   (a) Put all files (header file ended by .hpp and C++ source code ended by .cpp) in one directory.
   (b) Run the project after defining Hare.cpp, Tortoise.cpp, Road.cpp, Competition.cpp, and CompetitionTest.cpp.
      i. Approach 1: use g++ compile all source codes, then run ./a.out. Simple but not encourage.
      Reason: everytime a source code is modified, all the others source code are compiled and linked.
      To compile all files in this project, enter
      g++ *.cpp
      Run the project use
      ./a.out
      ii. Approach 2: use makefile.
      Copy the following `makefile` to the same folder where cpp and hpp files are located.

```makefile
# This is an example Makefile for a Hare and Tortoise Competition project.
# This program uses Hare, Tortoise, Road, and Competition modules.
# Typing 'make' or 'make run' will create the executable file.
#

# define some Makefile variables for the compiler and compiler flags
# to use Makefile variables later in the Makefile: $()
```

```
#
# -g    adds debugging information to the executable file
# -Wall turns on most, but not all, compiler warnings
#
# for C++ define CC = g++
CC = g++ -std=c++11
#CFLAGS = -g -Wall

# typing 'make' will invoke the first target entry in the file
# (in this case the default target entry)
# you can name this target entry anything, but "default" or "all"
# are the most commonly used names by convention
#
all: run

# To create the executable file run we need the object files
# CompetitionTest.o, Competition.o, Hare.o, Tortoise.o, and Road.o:
run:  CompetitionTest.o Competition.o Hare.o Tortoise.o Road.o
	$(CC) -o run CompetitionTest.o Competition.o Hare.o Tortoise.o Road.o

# To create the object file CompetitionTest.o, we need the source
# files CompetitionTest.cpp, Competition.h
CompetitionTest.o: CompetitionTest.cpp
	$(CC) -c CompetitionTest.cpp

# To create the object file Competition.o, we need the source
# files Competition.cpp and Competition.h
#
Competition.o: Competition.cpp
	$(CC) -c Competition.cpp

# To create the object file Hare.o, we need the source files
# Hare.cpp. By default, $(CC) -c Hare.cpp generate Hare.o
#
Hare.o: Hare.cpp
	$(CC) -c Hare.cpp

# To create the object file Tortoise.o, we need the source files
# Tortoise.cpp.
# By default, $(CC) -c Tortoise.cpp generates Tortoise.o
Tortoise.o: Tortoise.cpp
	$(CC) -c Tortoise.cpp

# To create the object file Road.o, we need the source files
# Road.cpp.
# By default, $(CC) -c Road.cpp generates Road.o
```

```
Road.o: Road.cpp
    $(CC) -c Road.cpp

# To start over from scratch, type 'make clean'. This
# removes the executable file, as well as old .o object
# files and *~ backup files:
#
clean:
    $(RM) run *.o *~
```

Afterwards, run

```
make
./run
```

2. Online editor gdb

   If you use online editor like `https://www.onlinegdb.com/`,

   (a) In the original main.cpp provided this editor, comments its contents using multi-line commentor /* before the first line and */ after the last line. Reason: main.cpp contains function main. A project cannot have more than one main function. Note that your project cannot have main.cpp, otherwise there would be name conflict in editor gdb.

   (b) Upload all your files using by click upload icon .

   (c) Click Run icon to run the project.

8 Sample outputs

Here is a sample output.

```
   1 H  T
   2 H    T
   3 H       T
   4       H    T
   5             H T
   6                T  H
   7                H T
   8                T
Ouch! Tortoise bites hare.
   9             HT
  10               T   H
  11                 T       H
  12                  T        H
  13                   T       H
  14                    T        H
  15                       T     H
  16                        T H
  17                         TH
  18                       T
Ouch! Tortoise bites hare.
  19                        TH
  20                         T
Ouch! Tortoise bites hare.
  21                       T H
  22                         T    H
Yuck. Hare wins.
```

Here is another sample run.

```
  1   T
Ouch, Tortoise bites Hare.
  2  T  H
  3    T  H
  4      TH
  5     T H
  6    T    H
  7      T  H
  8     T    H
Yuck, hare wins.
```

Here is yet another sample output.

```
  1 H T
  2    T  H
  3      TH
  4        TH
  5         T
Ouch, Tortoise bites Hare.
  6          T
Ouch, Tortoise bites Hare.
It is a tie.
```