CISS245: Advanced Programming
Quiz q6003

Name: cadalebout1@cougars.ccis.edu Score:

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

Q1. A hunter lives in the Wumpus world, which is a 4-by-4 grid. He/she walks in the direction he/she is facing. Initially he/she is facing N (north). In each time step in Wumpus world, the hunter (1) randomly selects a direction, (2) randomly decides to walk one step (in his/her direction) or not, and (3) randomly decides to fire one arrow (in his/her direction) or not. Here's a test run:

```
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
|H| | | |
+-+-+-+-+
Hunter direction: N

+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| |H| | |
+-+-+-+-+
Hunter direction: E

+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| |H| | |
```

```
+-+-+-+-+
Hunter direction: S

+-+-+-+-+
| | | | |
+-+-+-+-+
| |^| | |
+-+-+-+-+
| |H| | |
+-+-+-+-+
| | | | |
+-+-+-+-+
Hunter direction: N

+-+-+-+-+
| |^| | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | |H| |
+-+-+-+-+
| | | | |
+-+-+-+-+
Hunter direction: E

+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | |H|
+-+-+-+-+
| | | | |
+-+-+-+-+
Hunter direction: E
```

The ^ is the arrow flying in the N direction. You should draw the arrow using ^, v, >, < depending on the direction N,S,E,W (respectively) of flight.

Note that when the hunter fires his/her arrow, the arrow will fly in the direction of the hunter (at that point in time). The hunter can only fire one arrow (too bad). The hunter will also not walk into his arrow.

You are given this:

```
// file: WumpusWorld.h
#ifndef WUMPUSWORLD_H
#define WUMPUSWORLD_H
```

```
#include <iostream>

class WumpusWorld
{
public:
    init();              // initialize so that hunter is at (3,0).
    println();           // print according to the above format
    move_hunter();       // 1. randomize a direction rand() % 4 where 0,1,2,3
                         //    are N,S,E,W for hunter.
                         // 2. randomize whether to walk or not
                         // 3. randomize whether to fire arrow or not
                         // Of course hunter must stay in the world.
                         // If hunter tries to move N but its row is 0,
                         // hunter stays put.
    move_arrow();        // move arrow (if necessary)
private:
    char world_[4][4];
    int hunter_direction_;
    int arrow_state_;    // 0-N, 1-S, 2-E, 3-W, 4-not fired, 5-hit something
};
#endif
```

You want to make the hunter wonder around like this:

```
#include <iostream>
#include <ctime>
#include <cstdlib>
#include "WumpusWorld.h"

int main()
{
    srand((unsigned int) time(NULL));

    WumpusWorld ww;
    ww.init();
    ww.println();
    for (int i = 0; i < 50; ++i)
    {
        ww.move_hunter();
        ww.move_arrow();
        ww.println();
    }
    return 0;
}
```

Complete the following cpp file that contains the implementation of the methods declared in the WumpusWorld class.
Answer:

```
// file: WumpusWorld.cpp

void WumpusWorld::init()
{
    for(int i = 0; i < 4; ++i)
        for(int j = 0; j < 4; ++j)
        {
            world_[i][j] = ' ';
        }
    world_[3][0] = 'H';
}

void WumpusWorld::println()
{
    std::cout << "+-+-+-+-+\n";
    for(int i = 0; i < 4; ++i)
    {
        for(int j = 0; j < 4; ++j)
        {
            std::cout << '|' << world_[i][j];
        }
        std::cout << "|\n";
        std::cout << "+-+-+-+-+\n";
    }
}

void WumpusWorld::move_hunter()
{
    int dir = rand() % 4;
    if(rand() % 200 == 1 && arrow_state_ == 4)
    {
        arrow_state_ = dir;
    }
    if(rand() % 200 == 1)
    {
        switch(dir)
        {
            case 0:
                for(int i = 1; i < 4; ++i)
                {
                    for(int j = 0; j < 4; ++j)
                    {
                        if(world_[i][j] == 'H')
                        {
                            if(world_[i][j] == '^' ||
                               world_[i][j] == 'v' ||
                               world_[i][j] == '<' ||
                               world_[i][j] == '>')
                            {}
                            else
```

```
                    {
                        world_[i][j] = ' ';
                        world_[i-1][j] = 'H';
                    }
                }
            }
        }
        break;
    case 1:
        for(int i = 0; i < 3; ++i)
        {
            for(int j = 0; j < 4; ++j)
            {
                if(world_[i][j] == 'H')
                {
                    if(world_[i][j] == '^' ||
                       world_[i][j] == 'v' ||
                       world_[i][j] == '<' ||
                       world_[i][j] == '>')
                    {}
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i+1][j] = 'H';
                    }
                }
            }
        }
        break;
    case 3:
        for(int i = 0; i < 4; ++i)
        {
            for(int j = 1; j < 4; ++j)
            {
                if(world_[i][j] == 'H')
                {
                    if(world_[i][j] == '^' ||
                       world_[i][j] == 'v' ||
                       world_[i][j] == '<' ||
                       world_[i][j] == '>')
                    {}
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i][j-1] = 'H';
                    }
                }
            }
        }
        break;
    case 4:
```

```
                for(int i = 0; i < 4; ++i)
                {
                    for(int j = 0; j < 3; ++j)
                    {
                        if(world_[i][j] == 'H')
                        {
                            if(world_[i][j] == '^' ||
                               world_[i][j] == 'v' ||
                               world_[i][j] == '<' ||
                               world_[i][j] == '>')
                            {}
                            else
                            {
                                world_[i][j] = ' ';
                                world_[i][j+1] = 'H';
                            }
                        }
                    }
                }
                break;
        }
    }
}

void WumpusWorld::move_arrow()
{
    bool moved = false;
    switch(arrow_state_)
    {
        case 0://north
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; ++j)
            {
                if(world_[i][j] == '^' && moved = false)
                {
                    if(i == 0)
                    {
                        arrow_state_ = 4;
                        world[i][j] = ' ';
                        move = true;
                    }
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i-1][j] = '^';
                        move = true;
                    }
                }
            }
        }
```

```
        break;
        case 1://south
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; ++j)
            {
                if(world_[i][j] == 'v' && moved = false)
                {
                    if(i == 3)
                    {
                        arrow_state_ = 4;
                        world[i][j] = ' ';
                        move = true;
                    }
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i+1][j] = 'v';
                        move = true;
                    }
                }
            }
        }
        break;
        case 2:// east
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; ++j)
            {
                if(world_[i][j] == '<' && moved = false)
                {
                    if(j == 0)
                    {
                        arrow_state_ = 4;
                        world[i][j] = ' ';
                        move = true;
                    }
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i][j-1] = '<';
                        move = true;
                    }
                }
            }
        }
        break;
        case 3:
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; ++j)
```

```
            {
                if(world_[i][j] == '^' && moved = false)
                {
                    if(j == 3)
                    {
                        arrow_state_ = 4;
                        world[i][j] = ' ';
                        move = true;
                    }
                    else
                    {
                        world_[i][j] = ' ';
                        world_[i][j+1] = '>';
                        move = true;
                    }
                }
            }
        }
        break;
    }
}
```

Instructions

In `main.tex` change the email address in

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

to yours. In the bash shell, execute "`make`" to recompile `main.pdf`. Execute "`make v`" to view `main.pdf`. Execute "`make s`" to create `submit.tar.gz` for submission.

For each question, you'll see boxes for you to fill. You write your answers in `main.tex` file. For small boxes, if you see

```
1 + 1 = \answerbox{}.
```

you do this:

```
1 + 1 = \answerbox{2}.
```

`answerbox` will also appear in "true/false" and "multiple-choice" questions.

For longer answers that needs typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
int x;
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?
\begin{answerlong}
\end{answerlong}
```

you can write

```
What is the color of the sky?
\begin{answerlong}
The color of the sky is blue.
\end{answerlong}
```

For students beyond 245: You can put LaTeX commands in `answerbox` and `answerlong`.

A question that begins with "T or F or M" requires you to identify whether it is true or false, or meaningless. "Meaningless" means something's wrong with the statement and it is not well-defined. Something like "$1+_2$" or "$\{2\}^{\{3\}}$" is not well-defined. Therefore a question such as "Is $42 = 1+_2$ true or false?" or "Is $42 = \{2\}^{\{3\}}$ true or false?" does not make sense. "Is $P(42) = \{42\}$ true or false?" is meaningless because $P(X)$ is only defined if X is a set. For "Is $1 + 2 + 3$ true or false?", "$1 + 2 + 3$" is well–defined but as a "numerical expression", not as a "proposition", i.e., it cannot be true or false. Therefore "Is $1 + 2 + 3$ true or false?" is also not a well-defined question.

When writing results of computations, make sure it's simplified. For instance write 2 instead of $1 + 1$. When you write down sets, if the answer is $\{1\}$, I do not want to see $\{1, 1\}$.

When writing a counterexample, always write the simplest.

Here are some examples (see `instructions.tex` for details):

1. T or F or M: $1 + 1 = 2$ ..... T
2. T or F or M: $1 + 1 = 3$ ..... F
3. T or F or M: $1+^2 =$ ..... M
4. $1 + 2 =$ 3
5. Write a C++ statement to declare an integer variable named `x`.

   ```
   int x;
   ```

6. Solve $x^2 - 1 = 0$.

   Since $x^2 - 1 = (x - 1)(x + 1)$, $x^2 - 1 = 0$ implies $(x - 1)(x + 1) = 0$. Therefore $x - 1 = 0$ or $x = -1$. Hence $x = 1$ or $x = -1$.

7. Which is true? C
   (A) $1 + 1 = 0$
   (B) $1 + 1 = 1$
   (C) $1 + 1 = 2$
   (D) $1 + 1 = 3$
   (E) $1 + 1 = 4$