cisc1110, fall 2010, assignment for unit V

This is the assignment for unit V, "control structures." You are expected to complete the assignment in the C++ language and submit your ".cpp" file. You must complete and submit the assignment on or before the due date of **October 27**. *Remember: Assignments will NOT be accepted more than 7 days late.*
This assignment is worth **5 points**, plus 1 point of extra credit.

Submission instructions:

- Submit your assignment to me via email: `sklar@sci.brooklyn.cuny.edu`
- Your email subject line should be: **CISC 1110 Lab V submission**
- Attach your C++ (**sprite.cpp**) file to your email.
- Make sure your name is in the body of the email message.
- Make sure your name is also included in the header comments at the top of your C++ file.

Overview

For this assignment, you will create a program that simulates a sprite moving around in a 2-dimensional world. Assume that the size of the world is 10×10 grid cells. The sprite's location in its world is represented by (x, y) coordinate values, where x and y are each between 0 and 9.

The sprite game is played as follows. At the beginning of the game, the sprite's location is initialized to a random position in the 10×10 world. Also at the beginning of the game, a secret treasure is "placed" in the world; its location is also initialized to a random position in the 10×10 world. The player is shown the position of the sprite, but not the position of the treasure. The player has 10 turns to see if she can "move" the sprite around in its world and find the treasure. The sprite will "find" the treasure if its (x, y) location is the *same* as the treasure's (x, y) location. The player can enter the following "moves":

- $\mathbf{n}$ or $\mathbf{N}$ to go one grid cell to the north ($-y$ direction)
- $\mathbf{s}$ or $\mathbf{S}$ to go one grid cell to the south ($+y$ direction)
- $\mathbf{e}$ or $\mathbf{E}$ to go one grid cell to the east ($+x$ direction)
- $\mathbf{w}$ or $\mathbf{W}$ to go one grid cell to the west ($-x$ direction)
- $\mathbf{q}$ or $\mathbf{Q}$ to quit early (i.e., before finding the treasure and before using up all her moves)

Every time the player enters a move (besides "quit"), your program should increment the number of moves made by the player and update the location of the sprite. Then your program should check to see if the sprite has "found" the treasure. If the treasure has been found, then your program should display a "you won!" message to the player and exit the program. If the treasure has not been found, then your program should check to see if the player has used up all her moves. If she has used up all her moves, then your program should display a message saying "you lose :-(" and exit the program. Otherwise, the program should continue and let the player enter another move.

step 1 (1 point)

- Create a new program file named **sprite.cpp**.
- Initialize the random number generator.
- Declare two integers that will store the sprite's (x, y) location in the world.
- Declare two integers that will store the (x, y) location of the treasure in the world.
- Initialize the location of the sprite and the location of the treasure to random positions in the world. Make sure that both positions are within the $x = [0, 9]$ and $y = [0, 9]$ limits (inclusive; i.e., including 0 and 9).
- Compile, run and test your program. Go back and fix it if it doesn't work properly.
- *HINT:* You might want to print out both locations to make sure that you have initialized them correctly.

step 2 (1 point)

- Modify your program as follows.
- Declare a character variable that will store the player's move.
- Declare a boolean variable that will serve as the condition control for a `while` loop (below).
- Create a `while` loop that contains the following:
  - Displays the current location of the sprite.
  - Asks the player to enter her move (i.e., n, N, s, S, e, E, w, W, q, Q).
  - Branches based on the player's move and displays a message echoing the player's move (e.g., "the sprite will move north").
- In this step, don't worry about updating the sprite's position—just worry about printing the correct message based on what the player entered. This should include a message that says when the game is exiting and an error message if the player enters an invalid input.
- Compile, run and test your program. Go back and fix it if it doesn't work properly.

step 3 (1 point)

- Modify your program as follows.
- Declare an integer variable that will store the number of moves made by the player. Initialize it to 0.
- Before the player enters her move, tell her how many moves she has left.
- Every time the player enters a valid move, increment the number of moves.
- If the player reaches the maximum number of moves, then display a message and exit the program.
- Compile, run and test your program. Go back and fix it if it doesn't work properly.

step 4 (1 point)

- Modify your program as follows.
- Now go back to the part of your code where you displayed messages like "the sprite will move north" and change it so that the sprite's position is adjusted according to the player's move. Make sure that the sprite will not go outside the limits of its world (i.e., no position values < 0 or > 9).
- Here are some example test values for you to try out:
  - sprite starts at $(2, 2)$ and player enters $\mathbf{n} \Rightarrow$ sprite moves to $(2, 1)$
  - sprite starts at $(2, 2)$ and player enters $\mathbf{s} \Rightarrow$ sprite moves to $(2, 3)$
  - sprite starts at $(2, 2)$ and player enters $\mathbf{e} \Rightarrow$ sprite moves to $(3, 2)$
  - sprite starts at $(2, 2)$ and player enters $\mathbf{w} \Rightarrow$ sprite moves to $(1, 2)$
  - sprite starts at $(0, 0)$ and player enters $\mathbf{n} \Rightarrow$ sprite is at the north wall!
  - sprite starts at $(0, 0)$ and player enters $\mathbf{w} \Rightarrow$ sprite is at the west wall!
  - sprite starts at $(9, 9)$ and player enters $\mathbf{s} \Rightarrow$ sprite is at the south wall!
  - sprite starts at $(9, 9)$ and player enters $\mathbf{e} \Rightarrow$ sprite is at the east wall!
  - sprite starts at $(9, 9)$ and player enters $\mathbf{f} \Rightarrow$ invalid input. try again.
- Compile, run and test your program. Go back and fix it if it doesn't work properly.

step 5 (1 point)

- Modify your program as follows.
- If the player enters a valid move for the sprite, then check to see if the sprite has found the treasure. If it has, then display a message telling the player that she has won, and exit the game.
- Compile, run and test your program. Go back and fix it if it doesn't work properly.

extra credit step (1 extra credit point)

- Modify your program as follows.
- If the player enters a valid move for the sprite and the sprite has not found the treasure, then display a hint for the player to help her on her next move. If the player has moved closer to where the treasure is, then display "hotter!" If the player has moved further from where the treasure is, then display "colder!"
- *Hint:* Note that in order to complete this part, you will need to keep track of where the sprite is both *before* and *after* processing the player's move.
- Compile, run and test your program. Go back and fix it if it doesn't work properly.

sample output

Below is sample output from my solution to the assignment:

```
turn #1. you have 10 moves left.
the sprite is in position (3,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? n
the sprite will move north
turn #2. you have 9 moves left.
```

```
the sprite is in position (3,8).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
turn #3. you have 8 moves left.
the sprite is in position (4,8).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? w
the sprite will move west
turn #4. you have 7 moves left.
the sprite is in position (3,8).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? w
the sprite will move west
turn #5. you have 6 moves left.
the sprite is in position (2,8).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? w
the sprite will move west
turn #6. you have 5 moves left.
the sprite is in position (1,8).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? s
the sprite will move south
turn #7. you have 4 moves left.
the sprite is in position (1,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
turn #8. you have 3 moves left.
the sprite is in position (2,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
turn #9. you have 2 moves left.
the sprite is in position (3,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
turn #10. you have 1 moves left.
the sprite is in position (4,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
turn #11. you have 0 moves left.
the sprite is in position (5,9).
how would you like to move the sprite (n=north, s=south, e=east, w=west, q=quit)? e
the sprite will move east
game over. you lose :-(
```