Ideas for hangman

- Consider the class `Letters` and `Letters::GuessLetter`
  - Example: user trying to guess “hellfire”, initially `myDisplay` is “________” (no spaces)
  - If user guesses 'e', `myDisplay` should be “_ e ______ e”
  - Idea: build up a string one letter at a time

```cpp
def Letters::GuessLetter(const string& letter)
{
    string copy = "";
    for(k = 0; 
    {  if (myString.substr(k,1) == letter)
        {
            copy += letter;
        }
    }
    else ???
```

What about tracking letters?

- Idea one: store guessed letters in a string unless they’re already in the string

```cpp
bool Letters::AlreadyGuessed(const string& let)
// post: return true iff let has been guessed
{   if (myGuessedLetters.find(let) == string::npos){   return false;}return true;
```

- Initial value of `myGuessedLetters`? Where is this set? Where is it updated? Where is it defined?
- What about using a set?

When is game over?

- Game is over when user makes 10 misses
  - Where are misses tracked, who should count them (`Letters` or client program in `hang.cpp`?)
  - Who determines number of allowed misses: 6, 8, ... ?

- Game is over when user guesses the word (letter by letter)
  - Can this by determined in `Letters`?
  - If so, how does user find out if word is guessed? If not, why not?

- What does loop in `hang.cpp` do to figure out when game is over? What about playing again?

What about a Gallows class?

- You can define a class `Gallows`, what are its behaviors?
  - `DrawHead()` versus `AddPart()`
  - What about `DrawBody(int numPieces)`
  - Other alternatives? Advantages?

- Where is the `Gallows` object defined, in `hang.cpp`, in the class `Letters`? Other options?
  - Are there reasons to prefer putting the `Gallows` object `myGallows` in the `Letters` class? When will the hanged-person be drawn?
  - What about including the gallows object in `main` or in `hang.cpp`?