Self-Reference

Self-reference is one of the trickier topics from the tail end of the quarter. This series of questions explores self-reference through a series of questions about a variety of different programs.

i. What does the following program do?
   ```
   int main() {
     string input = getInput();
     string me = mySource();

     if (input == me) accept();
     else reject();
   }
   ```

In the proof from lecture we did that A\textsuperscript{TM} is undecidable, we began by assuming that A\textsuperscript{TM} was decidable. That meant there must be some decider for A\textsuperscript{TM}, which, in software, we represent as a method

   ```
   bool willAccept(string program, string input)
   ```

that takes as input the source code of a program and an input string, then returns true if the specified program will accept the specified input and returns false otherwise.

ii. Consider the following program:
   ```
   int main() {
     string input = getInput();

     if (input == 
```
Self-reference for decidability can be a tricky topic. If you haven't yet done so, you should pause and go read the Guide to Self-Reference on the course website.

Let's look at the self-referential program we wrote in lecture that we used to show $A_{TM}$ was undecidable:

```c
int main() {
    string me = mySource();
    string input = getInput();

    if (willAccept(me, input)) {
        reject();
    } else {
        accept();
    }
}
```

Let's go look at this code in some more detail.

vi. Suppose we feed the string `abba` as input to the above program. Explain why if `willAccept` says that the program accepts `abba`, then the program does not accept `abba`.

vii. Suppose we feed the string `abba` as input to the above program. Explain why if `willAccept` says that the does not accept `abba`, then it does accept `abba`.

viii. Explain why your answers to parts (vi) and (vii) collectively result in a contradiction that shows that $A_{TM}$ is undecidable.

In the Guide to Self-Reference, we showed another self-referential program we could have written that would also help us see that $A_{TM}$ is undecidable:

```c
int main() {
    string me = mySource();
    string input = getInput();

    if (willAccept(me, input)) {
        while (true) {
            // Do nothing
        }
    } else {
        accept();
    }
}
```

Let's go look at this code in some more detail.

ix. Suppose we feed the string `abba` as input to the above program. Explain why if `willAccept` says that the program accepts `abba`, then it does not accept `abba`.

x. Suppose we feed the string `abba` as input to the above program. Explain why if `willAccept` says that the program does not accept `abba`, then it does accept `abba`.

xi. Explain why your answers to parts (ix) and (x) collectively result in a contradiction that shows that $A_{TM}$ is undecidable.

(Continued on the next page)
In the Guide to Self-Reference, we showed a third self-referential program related to $A_{TM}$:

```c
int main() {
    string me = mySource();
    string input = getInput();
    if (willAccept(me, input)) {
        accept();
    } else {
        reject();
    }
}
```

Let's go look at this code in some more detail.

xii. Suppose we feed the string $abba$ as input to the above program. Explain why if `willAccept` says that the program accepts $abba$, then it does accept $abba$.

xiii. Suppose we feed the string $abba$ as input to the above program. Explain why if `willAccept` says that the program does not accept $abba$, then it does not accept $abba$.

xiv. Explain why your answers to parts (xii) and (xiii) do not prove that $A_{TM}$ is decidable.

Self-Reference and Decidability

Consider the language $L = \{ \langle M \rangle | M$ is a TM that accepts at least one string $\}$. This language is undecidable. Let's go see why this is.

i. Suppose for the sake of contradiction that $L \in R$. This means that we could write a function

```
bool acceptsAtLeastOneString(string program)
```

that accepts as input the source code of a program, then returns true if the program accepts at least one string and returns false otherwise. Write a self-referential program that uses this function to obtain a contradiction. As a hint, recall the general template for these sorts of programs: have the program ask whether it accepts at least one string, then have it do the opposite of whatever it determines it's supposed to do.

ii. Formalize your reasoning from part (ii) by writing a formal proof that $L \notin R$. To do so, follow the proof template from lecture: assume that $L \in R$, describe what that assumption entails, write a program that causes a contradiction, then explain why you get a contradiction in all cases.