In this homework, you will implement a spoken dialogue system using the CSLU Speech Toolkit and the Tcl programming language (see the course web page http://www.csee.ogi.edu/class/cse550 for information about the CSLU toolkit and Tcl/Tk programming language). The system will be limited in that the system will control the direction of the dialogue: the system will ask questions, and the user will answer them. The user's responses will be highly constrained and just single words or short phrases.

**Question 1**

Implement a spoken dialogue system for a banking application using the Rapid Application Developer in RAD. It should use the following finite state machine.

```
 s1
 user id?

 s2
 password?

 s3
 type of transaction?

 balance
 withdraw
 transfer

 s4
 account?

 s5
 balance is ...

 s6
 account?

 s7
 amount?

 s9
 from?

 s10
 to?

 s8
 balance is ...

 s11
 amount?

 s12
 balance in $s9(recog) is ... and $s10(recog) is...
```

It is sufficient if it only works for a single user id, say user 555, which can be hardcoded into state s1. The correct password can be hardcoded into s2. Assume the user has a checking, savings, and credit card account. Assume that the user has $100 in checking and $100 in savings, and owes $100 on the credit card. You can also assume that $10 and $20 are the only amounts that are ever withdrawn or transferred. The user should be able to run through this several types with the correct balances being maintained. Note that some states will need some tcl code. For example, one of your states will need to set the balance that the user has in...
each account. Note that this is being done with global variables.

```tcl
set ::balance(checking) 100
set ::balance(savings) 100
set ::balance(creditcard) 100
```

**Writeup 1.1** To what states did you add tcl code, and what was that code?

**Question 2**

Allow the user to withdraw any amount of money between $10 and $100 in multiples of $10.

**Writeup 2.2** What did you modify to make this work?

**Question 3**

Ensure that the user does not withdraw or transfer more money than they have in the account. An error state should be added that plays an appropriate error message. The user should be returned to the amount state so that they can re-enter the amount.

**Writeup 3.3** Writeup what you did.

**Question 4**

Due to speech recognition errors, sometimes the user will be misunderstood. For instance, the user might have said `withdraw` but the system thought they said `transfer`. The same thing could happen for the type account and money amount. Change your dialogue system so that it allows the user to recover from such mistakes.

**Writeup 4.4** Describe the functionality that you added. Will it always allow the user to recover from speech recognition errors? How much overhead does it add to the conversations?

**Question 5**

Sometimes users change their mind about what they want to do. Add in functionality so that the user can exit at any time.

**Writeup 5.5** Describe this functionality.

**Question 6**

Hardcoding a single userid and a single password as the speech recognition grammars for s1 and s2 will heavily bias the recognizer towards recognizing them (of course it does have a garbage state that forces it into an repair subdialogue). How can you improve the security of this system? Add this functionality to your system.

**Writeup 6.6** Writeup what you did.