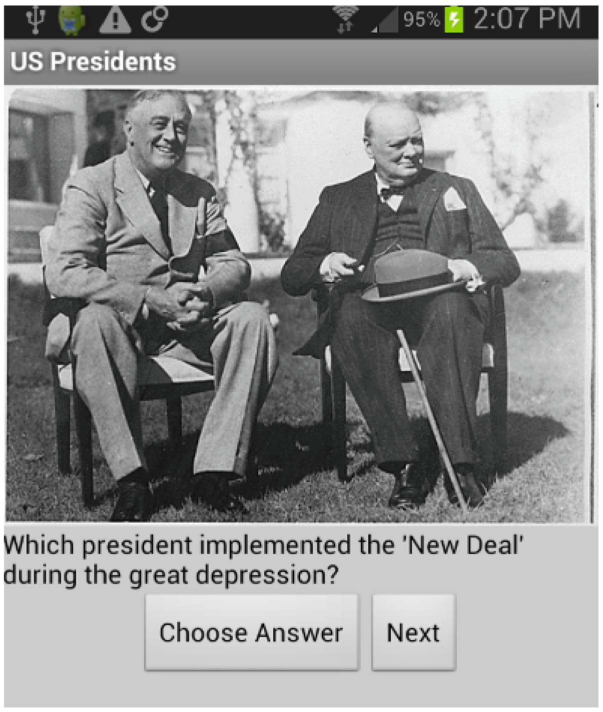
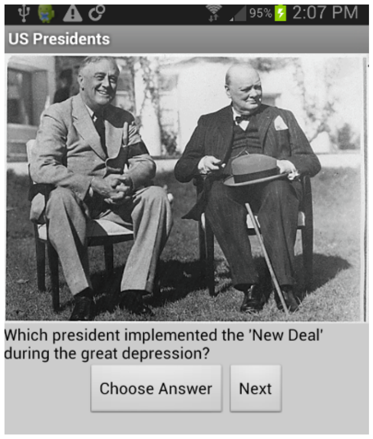
This version of the Presidents Quiz app introduces a couple of new features. One is the ***List Picker*** component, which you can see at work in the preview video. It provides the user with a list of options, letting us create *multiple-choice quizzes*, rather than just *short-answer quizzes* as in the previous version of the app. The other feature is a complex programming construct known as a ***list of lists*** -- i.e., a list that contain ***sublists*** as its elements. The *list of lists abstraction* is a ***data structure*** that you will find useful not only in this app but it lots of upcoming apps. **Objectives**: In this lesson you will learn how to: ● use the *ListPicker* component to provide choices to the user; ● define and use a new programming abstraction, a *list of lists*.	 US Presidents Which president implemented the 'New Deal' during the great depression? ***Click to watch preview video.***

Getting Ready

To get started, open App Inventor with the List of Lists template in a separate tab and follow along with the tutorial.

List of Lists Tutorial

The List of Lists UI


2:07 PM
US Presidents
Which president implemented the 'New Deal' during the great depression?
Choose Answer
Next

The *ListofListsTemplate* is the completed Presidents Quiz app. Everything should look as it did when you completed the Presidents Quiz tutorial.

Coding the Behavior

Recall that the Presidents Quiz app works like an interactive quiz. The user answers a series of questions about various U.S. presidents using a Textbox and Answer button. With each question the app reports whether the user's answer is correct or incorrect. The user can move on to the next question by clicking the Next button which causes a new question and its corresponding picture to appear. Questions, their correct answers, and pictures are stored in three separate lists. *Indexing* is used to make sure that a particular question is matched with its correct answer and correct picture.

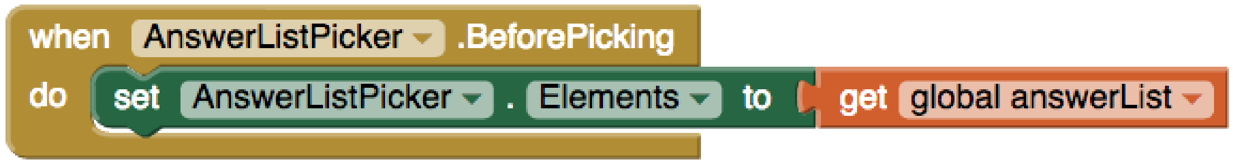
In this exercise, you'll change the app so that the user chooses an answer instead of typing it in. You'll first code it so that the answer choices come from the list of correct answers (answerList). Then you'll modify it so that there is a separate list of answer choices for each questions. For this, you'll define a variable answerChoices which is a list of lists.

Let the User Answer with a ListPicker

App Inventor provides the ListPicker component for allowing the user to choose from a list of options. In the Designer, remove the AnswerTextBox and AnswerButton from the user interface of your previous quiz and replace these with a ListPicker component. ListPicker has a button associated with it. When the user clicks the button, a list of choices is displayed. Set the Text property of the ListPicker so that the associated button reads “Choose Answer” as in the above picture.

To learn about ListPicker, you'll create a first version that allow the user to choose one of the three answers that are the answers for the first three questions. Later, you'll learn how to have different answer choices for each question using a list of lists.

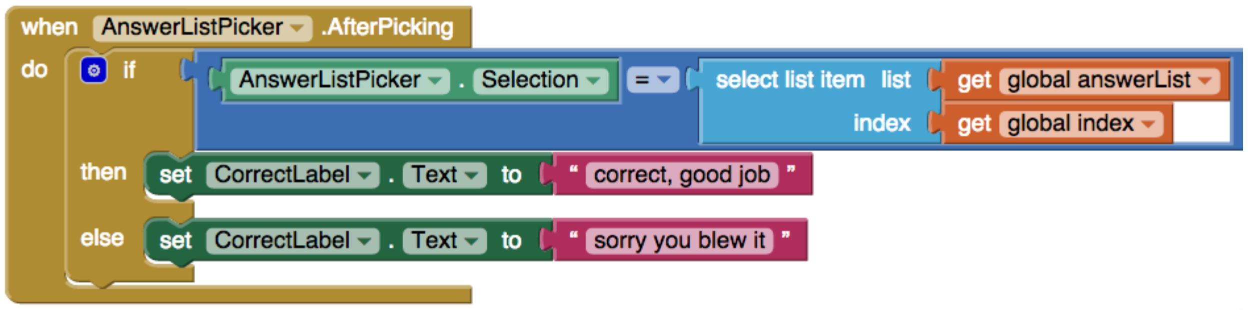
ListPicker has two important event handlers, BeforePicking and AfterPicking. BeforePicking is triggered when the user clicks the ListPicker button and before the choices are displayed. In BeforePicking, you specify which items will appear as choices by setting the ListPicker.Elements property. In this case, you'll set ListPicker.Elements to the answerList:



```
when AnswerListPicker .BeforePicking
do  set AnswerListPicker . Elements to  get global answerList
```

This will cause the ListPicker to appear with the three answers as choices ("Roosevelt", "Carter", "Nixon").

The ListPicker.AfterPicking event is used to do something once the user has chosen an answer. In this case, we just want to compare the choice with the right answer (using the index to choose the current answer from answerList):



```
when AnswerListPicker .AfterPicking
do  if    AnswerListPicker . Selection  =  select list item  list   get global answerList
                                                             index  get global index
    then  set CorrectLabel . Text to  " correct, good job "
    else  set CorrectLabel . Text to  " sorry you blew it "
```

Test your app at this point-- does it correctly evaluate the user's answers?

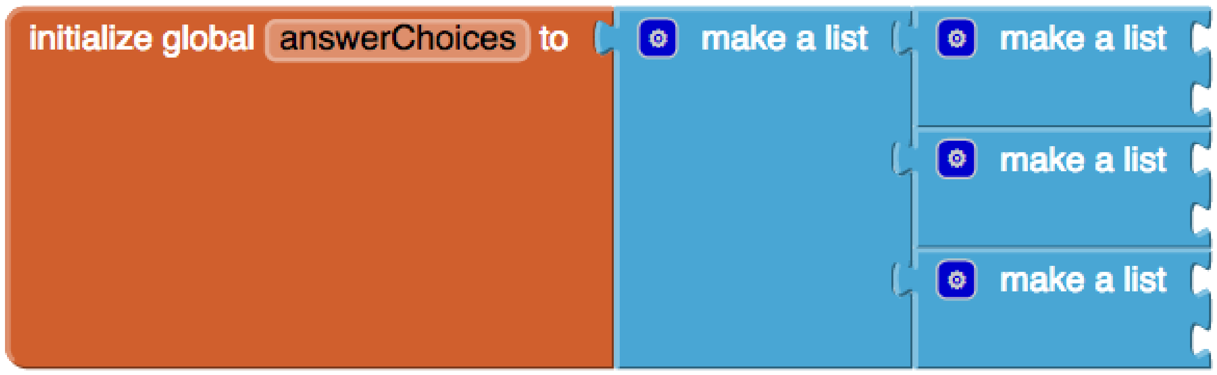
Provide Answer Choices for each Question

Let's make this app a little more challenging for the user by implementing what's called a *list of lists*. The items of a list can be numbers, text, or colors. But, the items in a list can also be lists. Hence, we call it a *list of lists.*

Begin by creating a new list.

1. Get a *initialize global variable* block and name the variable *answerChoices*
2. Make this variable a list variable by getting a *make a list* block.

Now you have a new list called *answerChoices* that is currently empty. In order to make this a list of lists, you will need another three *make a list* blocks, and you'll plug them in as the items of answerChoices

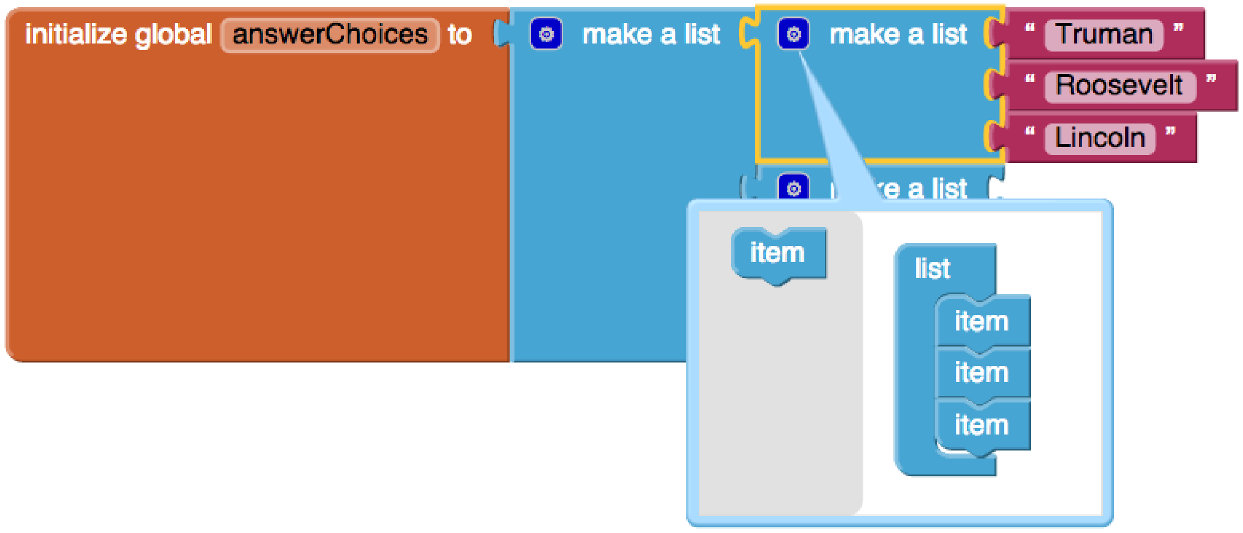


Now *answerChoices* is a list with three items. Each item is also a list which is currently empty. Since you have three questions in your quiz, you can have three sets of answer choices for the user to pick from (one set for each question). This can be done using *indexing*. You will also need to add more items to each of the lists.

Adding Items to a List of Lists

Start with the first index of *answerChoices*:

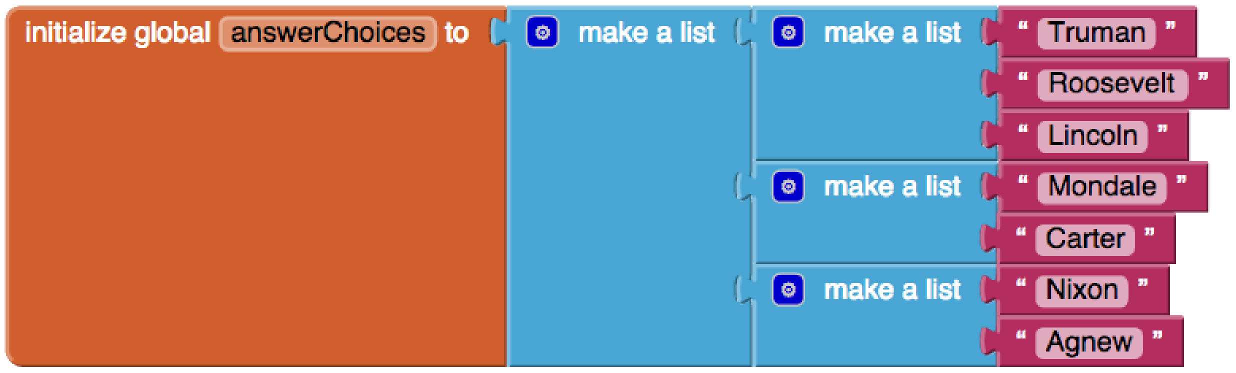
1. Click the blue plus sign of the second *make a list* block.
2. Drag one more item over to make the list have a total of three items. (You may choose to add more or less.)
3. Add in some answer choices, but be sure to have 'Roosevelt' as one of the choices.



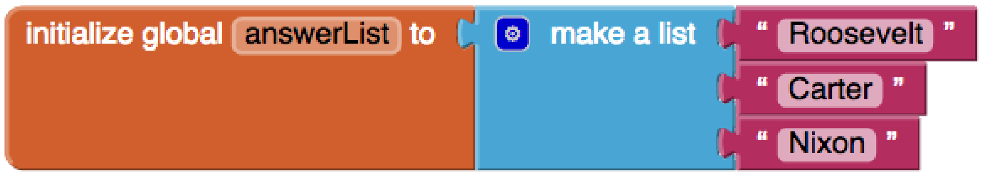
Now add more items to the other sublists of *answerChoices* so that there can be multiple answer choices for each question.

1. Click the blue plus sign of the *make a list* blocks to add additional items to each sublist.
2. Remember to always have the real answer be one of the choices at the appropriate index.

Your code should look something like the following:



```
initialize global answerChoices to  make a list  make a list  " Truman "
                                                              " Roosevelt "
                                                              " Lincoln "
                                                 make a list  " Mondale "
                                                              " Carter "
                                                 make a list  " Nixon "
                                                              " Agnew "
```

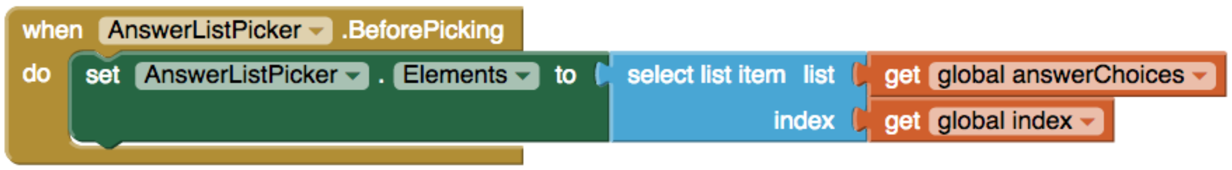


```
initialize global answerList to  make a list  " Roosevelt "
                                              " Carter "
                                              " Nixon "
```

The block of code on the bottom is included here to remind you of the correct answers. The correct answer must always be displayed in the list that is displayed to the user, or else their answer would always be incorrect. Recall, for example, that question 2's answer is the item at the second index in *answerList*, 'Carter'. The second index of *answerChoices* is a list of names that contains the correct answer 'Carter'. Next, we will program the app to display the list of choices at the *index* of *answerChoices* to the user.

Modify the List Picker's Before Picking Event Handler

Your *answerChoices* is a list of lists that now contains four separate lists. But, how will program know which list to display and when? Simple. This goes back to indexing and the global variable *index* that was created to keep track of the current question. Before the user selects their answer from the list picker, set the list picker's elements to be the list that is in *answerChoices* at the *index.* Your code should look like this:



```
when AnswerListPicker .BeforePicking
do  set AnswerListPicker . Elements to  select list item  list   get global answerChoices
                                                          index  get global index
```

Now you have a Presidents’ Quiz that is multiple choice with different possible answer choices appearing for each question. Now try some interactive exercises and reflect on what you learned in this lesson.

Nice work! Complete the Self-Check Exercises and Portfolio Reflection Questions as directed by your instructor.