How to Get an 'A' in Mr. Brooks' Computer Programming Class

Follow these simple guidelines and you're almost certain to get a fantastic grade in Mr. Brooks' class.

Pay Attention During Presentations (and Take Notes!)

There are three types of class days: presentation days, assignment days, and test days. On presentation days, Mr. Brooks will teach a new concept and show how to use that concept in a program. During the presentation, do these things:

- Take notes on the slides (these things will often show up on quizzes)
- Follow along with the code on your own computer
- Raise your hand and ask questions if something doesn't make sense

Watch the Class Web Page

The class web page is http://brooksprogramming.weebly.com. There are lots of valuable things there:

- This document
- Assignment and worksheet documents (in case you lose a handout)
- Study guides for tests (they are published when we get close to a test)

Work Hard During Assignment Time

On an assignment day, you'll have the whole class period to work on your programming assignment. This is valuable time, so don't waste it!

- If you get stuck, raise your hand and ask for help
- Feel free to bring headphones and listen to music (from your school computer)

Come in Before Class to Get Extra Help

If you need more help on an assignment that you're able to get during class time on assignment days, come in before class to get extra help. Not many students do this, so you'll get lots more one-on-one help during this time. I'm usually in the room as early as 7:15am. Send me an email telling me you're coming (and what time) and I will be there.

Re-take a Test If You Don't Like Your Score

The tests (two per term) make up a large portion of your grade. Make sure you get good scores for these. If you don't like your score you can retake the test (before school is best) and I'll take the better of the two scores (there's no risk).

Also, make sure you **get full credit for all assignments** by fixing any problems I tell you about before the due date.

Strategy for Writing a Computer Program

Writing a new program can be a bit daunting, whether it's a programming assignment in class or a program entirely of your own. Here is a simple strategy that can help you put it together.

Step 1: Read the Requirements

Read the program requirements very carefully. At least twice. Make sure you understand everything that the program is supposed to do. If this is a program you are inventing, take the time to draw pictures of what the windows will look like and what everything will do.

Step 2: Create a List of Tasks

From the program requirements, come up with a list of specific things that the program will need to do. It doesn't matter whether you know how to write code to do these tasks or not. It also doesn't matter what order the tasks are in. Just that the tasks are specific. (Some program requirements may produce several separate tasks in your list.)

Step 3: Map the Tasks to Events

From the program requirements (especially the picture of the window and all of its controls), identify all of the events that will occur in the program. These will be instances in the program lifetime where the program will need to do one or more of the tasks in your list. These will usually correspond to controls in your window, like button clicks. There will also be an event for the program startup.

Once you have a list of events, map each task from your list from step 2 to one or more events that will need to perform that task. If only a portion of a task will be performed in an event, break that task into sub-tasks and map the sub-tasks to events instead. As you are creating the map, you will also want to determine which order these tasks will be performed for each event.

Step 4: Break Up the Tasks into Smaller Tasks

If any of the tasks seems too complex or difficult for you to write the code for, break it up into smaller sub-tasks. Repeat the process until you're left with a (longer) list of simpler tasks that you know how to code. (If you don't know how to code a task and don't know how to break it up, ask for help.)

At the same time, identify any variables you'll need to implement the tasks and what the scope of those variables will need to be. Many tasks will share variables with other tasks. Most of the variables you identify here will be global scope, while variables you identify in the next step (while coding) will have a narrower scope.

Step 5: Write Code to Perform the Tasks

Now it's time to start coding. Create the window(s) for the program and add the necessary controls to them. Create event handlers for the events you have mapped tasks to above. In those event handlers, add a comment for each task for that event (in order). If you see that a particular sequence of tasks is showing up more than once, consider creating a method to perform those tasks.

Once all of the comments are in place, add the code to implement each task below the comment that describes the task. It doesn't really matter which ones you add code for first, but it's best to add code in such a way that you can run the program as you go and verify that the code is working.