General Hints

by AMG

General Hints

- 1) First, read the assignment document.

  Then read it again while you are analyzing the scaffold code you have been provided.

  Understand what you are expected to do.

  Then think about your approach. Top-down or down-top are both good. You may even have a mixture of these, but you should have a plan before writing your code. Do some math, logic and draw flow charts before coding if you need to (in fact, you usually need to.).

General Hints

- 2) Write small pieces at a time. Always keep your code free from compile errors. Write a small part, then compile, then if possible, run your code and check the output and see if anything is as you expected.

  Example: Code for 1 hour. 150 errors, 212 warnings. Spend 2 hours to get rid of all errors. Then run the program and see that your algorithm is wrong. Most of your efforts are wasted.

General Hints

- 3) Put debug lines at every possible place. This helps you to pinpoint the error easily. Read something from file? Who said you have read correctly? Print it to prove that you have read successfully!

  Debug lines work as checkpoints.

  (My suggestion is put //DEBUG at the end of those debug lines, thus when you finish your code you can search "DEBUG" to find your debug lines and comment them out. Do not delete them, just comment them out. People who read your code in the future may need those debug lines, including you.)

General Hints

- 4) Put comments in your code. Comments help people that read your code to understand your code easier. This may include future you. So when you go back to your code after a while, you may not understand even your own code.

  Example: This exactly happened to me years ago. I learned a good lesson from that.

General Hints

- 5) Be sure you know how to code with cout<< and cin>> properly.

  If you are using an external function, be sure how it works first. Then test it, then use it.