**Computer Programming Rubrics**

1. **Specifications/ Guideline (20%)**
2. A program must meet its **specifications** and **guideline**
3. Function correctly. This means that it behaves as desired, producing the correct output, for a variety of inputs.
4. **Logic /Programming Skills (30%)**
5. Logic helps organize code and enables programs to run more efficiently when implemented properly.
6. Programming logic is a set of principles that delineates how elements should be arranged so a computer can perform specific tasks.
7. **Visualizing**the logical relationships can help you see what the connections are among the different data points. Paper and pencil are often the better choice. There’s something about the physical act of drawing that helps keep a mind focused. Don’t worry about notations or conventions. This is a diagram for you to use as a way to develop your skills.
8. **Readability/ Coding Skills (50%)**
9. The code is exceptionally well organized and very easy to follow
10. **File comments** Each .cpp file should begin with a comment containing the file name, author name, date, and a description of the purpose of the code it contains.
11. All code should also be **well-commented**. This requires striking a balance between commenting everything, which adds a great deal of unneeded noise to the code, and commenting nothing, in which case the reader of the code (or you, when you come back to it later) has no assistance in understanding the more complex or less obvious sections of code. In general, aim to put a comment on any line of code that you might not understand yourself if you came back to it in a month without having thought about it in the interim. Using indentation consistently (e.g., every function's body is indented to the same level)
12. **Adding whitespace** (blank lines, spaces) where appropriate to help separate distinct parts of the code (e.g., space after commas in lists, blank lines between functions or between blocks of related lines within functions, etc.)
13. Giving **variables meaningful names**. Variables named A , B , and C or foo , bar , and baz give the reader no information whatsoever about their purpose or what information they may hold. Names like principal, maximum, and counter are much more useful. Loop variables are a common exception to this idea, and loop variables named i, j, etc. are okay.
14. No **irrelevant code**. Note: extra cin and cout will be extremely discouraging
15. The **code from internet sources** with some minor or major editing will be encouraging
16. **Extraordinary performance (Will be adjusted in sessional)**
17. Make a code for some engineering problems
18. Make a code for mathematical problems
19. Search similar problem on internet
20. Some practical problems search