# **ENGINEERING TOOLS – ENGR 110 FALL**

**PROJECT 1**

**PURPOSE:**  To develop, code, and execute a C++ program.

**DESCRIPTION**: Good data is important in analysis and decision making. Sometimes we design experiments to collect good data and sometimes we model situations to simulate data collection. Computers can help us model.

In this project, **you will develop a program that predicts the number of cars parked at HVCC at any given hour.** This information could be used by HVCC in many different ways - to identify traffic patterns, to budget for improved parking facilities, sidewalk planning and repair, to address safety issues, etc. To develop an accurate model that simulates actual parking at HVCC, we’d have to identify the factors that contribute to the number of cars parked on campus. That would require actual data collection. Instead, I’m going to provide you with this information (totally made up!). You then are responsible only for the development of the computer model (the source code). The information you need follows:

**The user will enter**

number of classes scheduled

number of tests scheduled that day

weather forecast

semester week

The calculation to determine how many cars we have on campus is performed as follows:

1. Always calculate the maximum number of cars that would be on campus using: maxcars = (number of classes scheduled \* 24 people per class)/1.02

2. Then we **adjust** the calculation accordingly:

1. Number of tests
2. Under 15 tests – no modification
3. Between 15 and 30 tests – subtract .15(24 \* number of classes scheduled)
4. More than 30 tests – subtract (24 \* number of classes scheduled)
5. Weather and semester week
6. Good weather
7. semester weeks 1-4: subtract (.12(number of classes) + Number of classes/1.02)
8. Semester weeks 5-11: subtract (.12(number of classes) + 50)
9. Semester weeks 12-16: subtract (Number of classes)
10. Bad weather
11. semester weeks 1-4: no modification
12. semester weeks 5-11: subtract (number of classes + (number of classes))/(.03(number of classes))
13. semester weeks 12-16: subtract (number of classes \* .05)

So, your final answer is calculated like this:

**FINAL ANSWER=MAXCARS – (adjustment for tests) – (adjustment for week and weather)**

**Your program should output all input values (number of classes scheduled, number of tests scheduled, weather forecast, semester week) and the final prediction calculated by the program. All output needs to be neatly labeled and presented.**

**DELIVERABLES**:

A. Your source code and sample output runs as follows:

Run 1: 325 classes scheduled, rainy weather, 3rd semester week, 17 tests scheduled.

Run 2: 77 classes scheduled, good weather, 12 semester week, 31 tests scheduled.

Run 3: 112 classes scheduled, good weather, 7 semester week, 5 tests scheduled.

Run 4: 501 classes scheduled, rainy weather, 10th semester week, 276 tests scheduled.

Run 5: 0 classes scheduled, rainy weather, 1st semester week, 0 tests scheduled.

B. Problem statement for this program (two or three sentences describing what your program does, list of inputs, list of outputs).

C. Flowchart which documents your algorithm for this program – this can be done with pencil, paper, AND A STRAIGHT EDGE. Or, you can use software such as Microsoft Visio or SMARTDRAW. An online version of this software can be found here: <https://www.smartdraw.com>. Click on the START NOW button. Either way, it MUST be neat.

All of these should be stapled together and handed in no later than the due date.

**EVALUATION:**

\*\*\* Problem statement - based on completeness and accuracy.

\*\*\* Flow chart - based on accuracy (does it reflect your program and is it complete), detail (is it in enough detail that it reflects your program and the tasks in the program), and presentation (is it neatly done, flows top to bottom, I can read it, etc).

\*\*\* Program - based upon the characteristics of good program design that we discussed in class. I will be looking especially for good programming style (meaningful comments, descriptive variable names, use of blank lines to organize the sections of your program, etc); nicely formatted, meaningful output; functionality (has all the parts); reliability (correct answers, simple error checks); user friendliness (clear instructions to user, nicely formatted output); and efficient code;

**NO LATE PROJECTS WILL BE ACCEPTED**.

No Exceptions.