Submitting Programs to the Automated Grader
Student's Guide

Virginia Polytechnic Institute and State University
Department of Computer Science

Table of Contents

Disclaimer: Every effort has been made to ensure that the contents of this document are accurate and complete. However, the Automated Grader Project is ongoing. It is always possible improvements and bug fixes have been implemented since the last update of this document. The current version of this document will be available from the New Automated Grader Homepage:
http://ei.cs.vt.edu/~cs1044/Grader/Grader.html

Introduction

The programming assignments for the course will be tested and scored by the New Automated Grader (NAG). This document describes how to submit your programs for grading, how the grading is performed, and how to interpret the results when trying to fix your mistakes.

The NAG is a client-server application written in the Java programming language. You will install the (free) client software, called *Submit*, on your computer and use that software to submit your programs. The installation process is described in the next section. When you submit a program, the client software will establish a network connection to the server software (usually called the Grader Server, or simply the Grader), and transfer your submission to the computer on which the Grader is running. After testing and scoring your program, the Grader Server will send you an e-mail message containing your score for that submission, and additional information. See the sections following the installation instructions for details and examples.

Installation

Using the client software, *Submit*, requires a computer running Windows 95/98 or Windows NT (it will not run on DOS or Windows 3.1) and a 32-bit connection to the Internet. In order to operate *Submit*, you must first obtain and install the Java Runtime Environment, a free software package from Sun Microsystems, Inc.

▪ Downloading the Java Runtime Environment JRE from the Internet:

1) Using a web browser (such as Netscape or Microsoft Explorer), connect to the Java website at **www.javasoft.com**
2) Click on the link for Products & APIs in the left-hand column.
3) Scroll down click on the Java Runtime Environment - JRE 1.1.7B link.
4) Click on the JRE 1.1 Win32 Release link. Click on the Continue button. Click the Agree button on the Export Terms and Control page.
5) Click on the `FTP download jre117B-win32.exe` button to start the download process. In the Save dialog box, select a folder and click the Save button. This will copy the file `jre117B-win32.exe` to the location you just selected; this file is a self-extracting archive. Be sure to remember the location you chose.

The Java Runtime will also be mirrored from the Grader Home page, so you may also obtain it when you download the Grader Submit Client as described below. Note, the Submit Client will also work with versions 1.1.3-1.1.6 of the Java Runtime.

▪ Installing the Java Runtime Environment JRE on your computer:

1) Installation in the Windows environment is straightforward. Using the Windows Explorer, find the folder where you saved the Java Runtime file, `jre117B-win32.exe`, and double-click on the file. This will launch an installation application.
2) Unless you want to customize the location the Java Runtime will be installed to, just click the Next button on each dialog box until the installation is complete. Note the location in which the Java runtime was installed; e.g., the default location would be `C:\jre1.1.7B`.
3) Now you must add the location of the Java runtime executable to your system path. The instructions given here assume this is the default location (shown above). If not, use the correct location when you edit the path.

For Win95/98:

- Use Windows Explorer to look in the root directory on your C drive.
- There should be a file named `autoexec.bat`
- Open that file in an editor (Notepad will do) and find the line that starts with "`path=`".
- Go to the end of that line and type a semicolon, followed by `C:\jre1.1.7B\bin`.

For WinNT:

- Go to the Start menu, to Settings, and start Control Panel.
- Open the System tool and click on the Environment tab.
- In the top window, scroll if necessary, and find the line starting with Path.
- Click on that line and the path string will be displayed in the text box line labeled "value" near the bottom.
- Edit that line to type a semicolon, followed by `C:\jre1.1.7B\bin`.
- Click on the Apply button and close the System tool and Control Panel.

▪ Downloading *Submit* from the Internet:

1) Using a web browser (such as Netscape or Microsoft Explorer), connect to the Automated Grader Home Page at **http://ei.cs.vt.edu/~cs1044/Grader/Grader.html**. Scroll down to the section for downloading the Client.

2) Select the appropriate version of *Submit* for your course and click on the link to download it.

3) When you are asked to choose a location to save the file, choose any location you like, but be sure to remember where you saved it. The self-extracting archive for *Submit* will be copied to your computer.

▪ Installing *Submit* on your computer:

1) Using the Windows Explorer, find the folder where you saved the *Submit* archive file, and double-click on the file. This will extract the contents of the archive file to your computer, creating a number of subfolders as well. Do not delete or alter any of these files. By default, *Submit* will be installed in a folder named `CSubmit` or `F90Submit` on the root of the current drive; you can change that during the extraction process if you like.

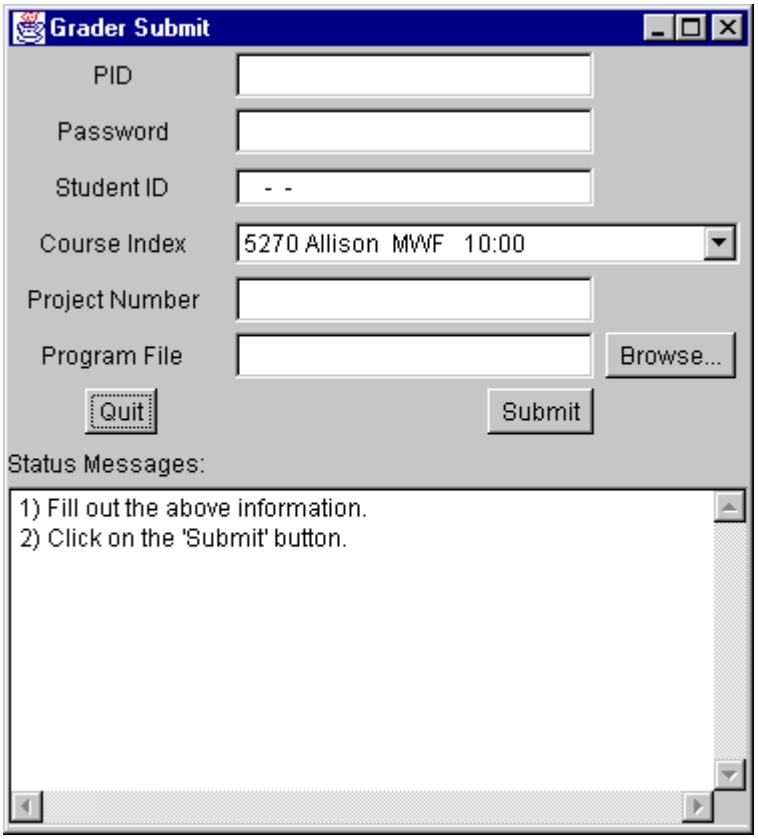
2) You can run *Submit* by double-clicking on the file `Submit.bat` in the `CSubmit` (or `F90Submit`) folder. To make it easier, you can create a shortcut on your desktop. Right-click (click and hold down the right mouse button) on the file `Submit.bat` and drag it to the desktop. When you release the mouse, a menu will pop up; select Create Shortcut(s) Here and you can then run *Submit* by double-clicking on the resulting shortcut icon.

Submitting a Program from Your Computer

This section describes how to use *Submit* to send in a program for automated grading. You can follow these steps after *Submit* has been installed onto your computer (see **Installation** above), or from one of the Windows NT computers in the Computer Science Undergraduate Laboratory in McBryde 116-118.

1. **Connect to the Internet**: *Submit* is designed to work over the Internet. Before you can use it, you must first run your communications software and connect to the network.

2. **Starting Submit:** To run *Submit* from Windows NT or Windows 95/98, go to Windows Explorer and find the folder into which *Submit* has been installed, then double-click on `Submit.bat`. Alternatively, if a desktop icon has been created for *Submit*, just double-click on that. You will be presented with a window similar to:

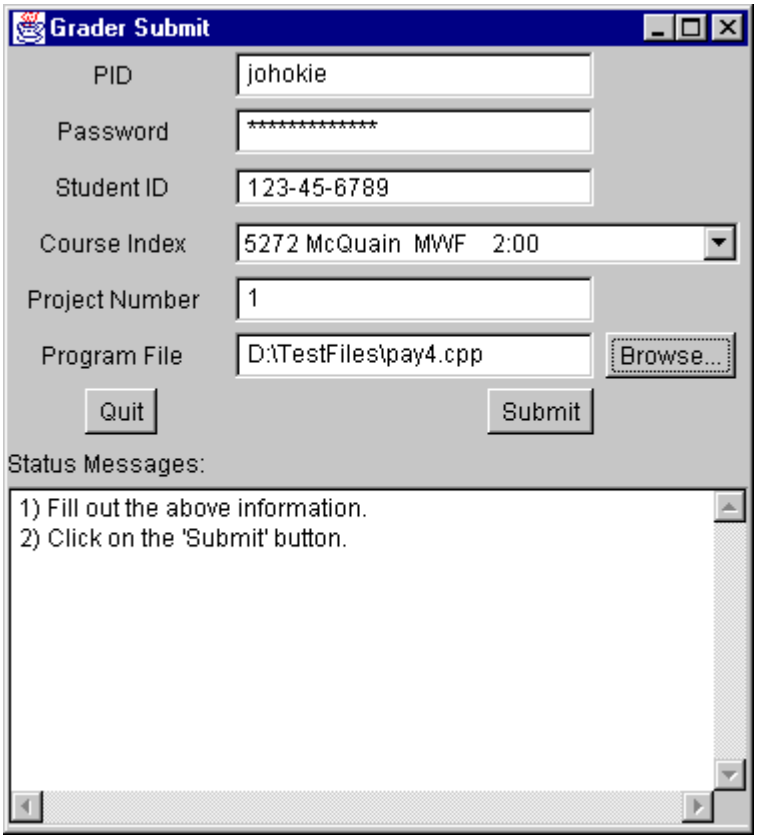


3. **Filling Out the Submission Form:** The window depicts a form that needs to be filled out in order for you to make a submission. You must fill out all of the information (PID, Password, student ID, class index, project number, and program file name) or *Submit* will not allow you to send your program. Use the arrow keys or the mouse to move between fields — the tab key is not interpreted properly. Each of the fields in the form is discussed in detail below:

   - **PID**: Enter your university PID (this is the same name you use when you connect to the university network or check your e-mail). This must be your original PID, not an e-mail alias. (Note: don't include `@vt.edu`)
   - **Password:** Type in your PID password (this would be the same password you use to connect to the university network or check your e-mail). As you type, each letter of your password will appear as an asterisk (*). This is done to hide your password from the sight of anyone who might be watching over your shoulder. Be careful that you type it correctly.
   - **Student ID:** Your social security number.

- **Course Index:** Click your mouse inside this field to get a list of class sections and index numbers. Make sure you select the correct time, instructor, and index number for the class you are enrolled in, or your program will be rejected.
- **Project Number:** This is the number of the programming assignment. For example, if you are making a submission for the third programming project, enter "3" here. This is *not* the number of submission attempts you have made for this assignment. Choosing the wrong number may result in the rejection of your submission, or in your getting a very low score for the wrong assignment.
- **Program File:** Fill in the file name for your program. You can also select the "Browse" button to visually search for the file. Be sure you send the correct file. If not, you'll waste a submission and receive a zero.

The resulting window should look something like:



4. **Submitting:** Once all of the above information has been entered, click on the "Submit" button. This will send in your program for automated grading, provided that all of the information you typed in is correct. Four things will happen when you press the "Submit" button:

   A. Your PID and password are checked for authenticity. If you entered your PID or password incorrectly, you will get a dialog box with an appropriate message and be asked to check the spelling of your PID or re-enter your password. For example:

   This check is done using the Virginia Tech campus e-mail server. Occasionally you may get a dialog box indicating that the Authentication Server is unavailable:

   If this happens, check your network connection (for example, try viewing the Grader web page). If that works, the e-mail server is probably either down or refusing connections because the system load is too high. Wait a few minutes and try again.

   B. *Submit* connects to the automated Grader server. If you get a message saying a connection can not be made:

   check that your communications software is running and is connected to the university network. If your computer is on the network and you still get a connection error, the Grader server or some other university network device might be down. Try again later.

   There is one other known cause of a connection failure at this point. If you are using a 16-bit network connection, such as the 16-bit Trumpet Winsock provided with the old WNet software, then *Submit* will not operate correctly. In this case you will have to either submit your program from the Computer Science Undergraduate Lab or switch to a 32-bit connection; the simplest solution is to obtain and install the current VTNet software package (after all, it's free).

C. Once a connection has been made, *Submit* will send your ID number, PID, section number and project number to the Grader server. The Grader will check to see if your ID number and PID match the roll file for the specified section. If they don't, you will receive an e-mail message from the Grader indicating the problem and your submission will be rejected. If everything checks out properly, *Submit* will send your program over the network to the Grader server where it will be automatically graded.

D. After the Grader server has received your program, it will disconnect from *Submit*, which will display the dialog box below:

At this point your submission is complete, and you can quit *Submit* by clicking on the Close "x" button in the upper right-hand corner of the *Submit* window or the "Quit" button.

5. Getting Results: The Grader server will automatically run your program, test it, and compute a score. It will then send you an e-mail message containing your score, the input data used for grading, the output generated by your program, and what the correct output should look like. If your program runs correctly, the right output should match your program's output. See the section How the Grader Scores Your Submission for further discussion. Under normal conditions, the e-mail message should be sent within a few minutes.

How the Grader Scores Your Submission

A sample e-mail message from the Grader Server is shown on page 10.

- **Submission Information:** The message tells you which submission this scoring was for and how many submissions are allowed for this project.
- **Point Information:** Provides your score, with a breakdown indicating any bonus or penalty points assessed. Your Instructor will determine how bonus and/or penalty points are to be assigned.
- **Input File:** Shows the input file on which this submission was tested.
- **Correct Output File:** The output file produced by the Instructor's solution using the preceding input file.
- **Student Output File:** The output file your submission produced from the same input file.

The Grader compares your output file to the correct output file, line by line. Each line is broken into "tokens" consisting of characters separated by whitespace (blanks or tabs). The tokens from each line of your output file are compared to those from the corresponding line of the correct output file; your score for each line is determined by the number of correct tokens you produce and the number of points assigned to that line. The point value assigned for each line is indicated by the number in square brackets at the beginning of each line of the correct output file.

Note that the amount of horizontal space you put between tokens does not matter to the Grader, unless you manage to run two tokens together so the Grader treats them as a single token (for example, printing "GrossPay" instead of "Gross Pay").

It is important that your output file not contain any extra lines, or omit any lines. If you do have extra lines or missing lines, then the Grader may compare the wrong lines and you will receive a very low score. For example, if Joe Bob Hokie had omitted the line of equal signs following the line of column labels in the output file, every line after the column labels line would have received a score of zero!

Explanation of scoring of a "good try": So why did Joe Bob get 61 for this submission? Compare the correct output file with Joe Bob's output. It's pretty obvious that Joe Bob doesn't have the correct averages for the insurance, income tax or net pay columns. Comparing the numbers in those columns with the correct answers, you find that Joe Bob has 4 of the insurance amounts wrong, and that all 6 of the income tax amounts are wrong. So, of course, all 6 of the net pay amounts are also wrong. Now each line in the body of the table is worth 10 points, so each amount (token) is worth 10/6 points. Joe Bob also has three of the averages wrong, and each of those is worth 24/6 points. If you add it all up, the Grader deducted 16*(10/6) + 3*(24/6) points. That works out to a deduction of about 38.67 points for a score of 61.3 — the Grader rounds the score to an integer, so Joe Bob got a 61 for this submission.

It is also vital that your labels match those used in the correct output file exactly. Otherwise those tokens will not match and the Grader will deduct points. Joe Bob modified the program to fix the calculations that were wrong before and resubmitted. Page 11 shows the Grader's response for the revised submission.

The reason Joe Bob received an 87 instead of 100 on this submission is that some of the labels are wrong now. The title on the second line is wrong, costing 2 points (Payroll is compared to March, so it's wrong, and he has one extra token). He also got some of the column headers wrong. Each token in that line is worth 12/8 = 1.5 points. Joe Bob got careless and omitted the word Pay, which threw the token comparison out of order. Each of Job Bob's remaining tokens failed the comparison, making 5 wrong tokens; Joe Bob also had two fewer tokens than the correct line and that was counted as two more wrong tokens. So the column headers cost Joe Bob a total of 10.5 points. So his score was 87.5, rounded to 88 by the Grader.

Extra or missing lines: Notice that some of the lines are given a value of 0 points. For instance, the first line is supposed to contain the name of the programmer. "Joe Bob Hokie" doesn't match the line in the correct output, but that's OK since the point value of the line is 0. That means that you don't have to match anything on that line exactly; but remember, if you don't have a line there it will throw the comparison off again.

Also, notice the horizontal lines that separate sections of the e-mail message. The line at the very end, after the student output section may help you determine your mistakes, in some cases. If your output file has extra nonblank lines, when compared to the correct output file then you will be penalized for each of those; the going rate for extra nonblank lines is 10 points. Compare the line counts given in the email message to see if you have too many or too few lines of output.

On the other hand, if your output has fewer lines than the correct output, then you're penalized the number of points that are assigned to those lines. For example, if Joe Bob had failed to print the line of averages after the body of the table, the Grader would have deducted 24 points.

How extra or missing blank lines are handled depends on where they occur. Extra blank lines at the end of your output file should be ignored by the Grader. If you insert an extra blank line in the middle of your output file, the Grader will assess a penalty for that, determined by your instructor, and then attempt to resynchronize its comparison by reading the next line of your output. If you omit a specified blank line, that is handled in a similar manner. Note that the email header specifies how many mismatches there were involving blank lines and the total penalty assessed for them.

Other messages, other problems: The two example Grader e-mail messages discussed above are the most common sort. However, there are several other scenarios that you should be aware of. If your submission does not compile (without errors) you will receive the following message:

```
Your source code failed to compile.
Please correct your code so that it compiles and then submit it again.
Your score for this submission is ZERO.
```

Of course, you should never submit a program that does not compile since all that does is waste a submission. Note two possible causes of this are using a different compiler (the Grader uses Visual C++ 5.0) or submitting the wrong file. There have even been instances where students have submitted term papers to the Grader – you shouldn't be surprised that a term paper usually doesn't compile properly.

The Grader gives your program a fixed amount of time to finish (the default is 10 seconds). If your program has not finished within the time limit, the Grader kills your program and you'll receive the message:

```
Your source code failed to exit and took too long to execute.
Possible problems include:
     - an infinite loop
     - a runtime error such as an array index out of bounds
       or divide by zero
     - your program expects input from the keyboard
Please correct your code so that it terminates properly.
Your score for this submission is ZERO.
```

When this happens, you need to do further testing and fix the problem before resubmitting. There is absolutely no point in resubmitting the same source code. This sort of problem may occur for a variety of reasons, including but not limited to the ones listed in the Grader message. The message will also include the input file and correct output, so you may use that input file to try to determine why your program misbehaved.

It is also possible that your program may be terminated abnormally (killed) by the operating system on the Grader machine before the time limit has expired. In that case, the Grader doesn't actually "know" that happened. It will look for an output file to score as usual and you will receive an e-mail message similar to the first one discussed above. However, if this happens you may notice that your output is incomplete since your program did not run to a normal termination. If that happens, you need to debug your program to eliminate the error before using another submission.

You may also find that when you run your program on your computer with the input file the Grader used, your program produces a complete output file and appears to operate correctly. If the Grader indicates your program is not producing

correct results, but it does something different on your computer, the most likely explanation is that you are testing your program under Windows 95. The Grader operates on a machine using Windows NT, which is much less forgiving of misbehaving programs that Windows 95 is. You may not be able to discover the source of the problem and fix it under Windows 95. In that case, you should test your program in the Computer Science Undergraduate Lab on one of the computers equipped with Windows NT. Understand: the fact that your program appears to run correctly under Windows 95 but not under Windows NT indicates only that Windows 95 tolerates bad behavior, not that your program is correct.

You may also get an e-mail message similar to:

```
Your source code failed to produce the output file: payola.dat
Possible problems include:
     - you specified the wrong output file name
     - you specified the wrong input file name
     - your program had a runtime error and was terminated
           by the system
Please correct your code so that it produces this output file.
Your score for this submission is ZERO.
```

In this case, the Grader couldn't find your output file. The correct name for your output file will be given in the specification for your assignment. If you use another name, the Grader will not find your output. If your program is terminated by the system because of a runtime error, before it produces any output, you'll see the same message. Again, test and determine what the problem is, and fix it, before using another submission.

The moral of all this is simple:

Follow the project specifications for output precisely!

Date: Tue, 18 Aug 1998 14:27:47 -0400 (EDT)
From: grader@vt.edu
Subject: C++ Grader Notification

PLEASE DO NOT REPLY TO THIS EMAIL MESSAGE

This information is for submit number 2 of 3 allowed submits.

Point Information

```
      Raw Score          : 61
      Early Bonus        : 2
      Late Penalty       : 0
      Blank Line Penalty : 0   (0 mismatches)
      ---------------
      Total Score   : 63
```

Input File

```
    1912      64   1957.00    B
    2179      35   4282.00    B
    6008      54   3324.00    B
    9171      59   1026.00    D
    5237      43   2318.00    D
    5131      36   3621.00    B
```

Correct Output File: 13 lines

```
[ 0]Bill McQuain
[ 4]Macro$oft Corporation Payroll
[ 0]
[12] IdNum   Gross Pay      Ins.    F.I.T.       SSI     Net Pay
[ 0]============================================================
[10]  1912     1957.00    150.00    547.96    132.10     1126.94
[10]  2179     4282.00    100.00   1413.06    289.04     2479.91
[10]  6008     3324.00    150.00   1096.92    224.37     1852.71
[10]  9171     1026.00    200.00    287.28     69.26      469.46
[10]  5237     2318.00    200.00    649.04    156.47     1312.49
[10]  5131     3621.00    150.00   1194.93    244.42     2031.65
[ 0]============================================================
[24]  Avg:     2754.67    158.33    864.86    185.94     1545.53
```

Student Output File: 13 lines

```
Joe Bob Hokie
Macro$oft Corporation Payroll

 IdNum     Gross Pay      Ins.    F.I.T.       SSI       Net Pay
============================================================
  1912       1957.00    175.00    489.25    132.10       1160.65
  2179       4282.00    125.00   1498.70    289.04       2369.27
  6008       3324.00    175.00   1163.40    224.37       1761.23
  9171       1026.00    200.00    256.50     69.26        500.24
  5237       2318.00    200.00    579.50    156.47       1382.04
  5131       3621.00    175.00   1267.35    244.42       1934.23
============================================================
  Avg:       2754.67    175.00    875.78    185.94       1517.94
```

```
Date: Tue, 18 Aug 1998 14:35:49 -0400 (EDT)
From: grader@vt.edu
Subject: C++ Grader Notification

PLEASE DO NOT REPLY TO THIS EMAIL MESSAGE

This information is for submit number 3 of 5 allowed submits.

_____________________________________________________________
Point Information
_____________________________________________________________
      Raw Score          : 88
      Early Bonus        : 2
      Late Penalty       : 0
      Blank Line Penalty : 0   (0 mismatches)
      ---------------
      Total Score   : 90
_____________________________________________________________
Input File
_____________________________________________________________
    5300      23    602.00    D
    3319      47   2359.00    B
    9760      62    986.00    B
    3876      68   2146.00    D
    6067      23   3481.00    D
    1607      19   4953.00    B
    1980      44   2194.00    D
    8079      52    775.00    D

_____________________________________________________________
Correct Output File: 15 lines
_____________________________________________________________
[ 0]Bill McQuain
[ 4]Macro$oft Corporation Payroll
[ 0]
[12] IdNum   Gross Pay      Ins.    F.I.T.       SSI     Net Pay
[ 0]============================================================
[ 7]  5300      602.00    140.00     90.30     40.64      331.07
[ 7]  3319     2359.00    150.00    660.52    159.23     1389.25
[ 7]  9760      986.00    150.00    147.90     66.56      621.54
[ 7]  3876     2146.00    300.00    600.88    144.86     1100.27
[ 7]  6067     3481.00    140.00   1148.73    234.97     1957.30
[ 7]  1607     4953.00    100.00   1634.49    334.33     2884.18
[ 7]  1980     2194.00    200.00    614.32    148.10     1231.58
[ 7]  8079      775.00    200.00    116.25     52.31      406.44
[ 0]============================================================
[28]  Avg:     2187.00    172.50    626.67    147.62     1240.20

_____________________________________________________________
Student Output File: 15 lines
_____________________________________________________________
Joe Bob Hokie
Macro$oft Corporation March Payroll

 IdNum     Gross           Ins     FIT          SSI       Net
================================================================
  5300       602.00    140.00     90.30     40.64      331.07
  3319      2359.00    150.00    660.52    159.23     1389.25
  9760       986.00    150.00    147.90     66.56      621.54
  3876      2146.00    300.00    600.88    144.86     1100.27
  6067      3481.00    140.00   1148.73    234.97     1957.30
  1607      4953.00    100.00   1634.49    334.33     2884.18
  1980      2194.00    200.00    614.32    148.10     1231.58
  8079       775.00    200.00    116.25     52.31      406.44
================================================================
  Avg:      2187.00    172.50    626.67    147.62     1240.20

_____________________________________________________________
```

Multiple Submissions

Some instructors allow programming assignments to be submitted more than once. This gives students a chance to fix problems that were detected by the Grader. The number of submissions that are allowed depends on your instructor. The Grader server will automatically ignore any submissions you send beyond the maximum number allowed by your teacher. How grades are assigned to multiple submissions are also up to the instructor. Some will use the highest grade from all of your submissions. Others may use the grade from your last submission.

If multiple submissions are allowed, the Grader will use a different input file each time. These input files will be generated by a program provided by your Instructor, and will conform to the program specification provided for your assignment.

You should send another submission only after you have received the results from the previous submission, determined what errors you had in that version of your program, and attempted to fix those errors.

Tips on Using Multiple Submissions Effectively

Here are a few tips that can help you use multiple submission effectively (if your instructor allows multiple submissions). Following these hints should help you get the most out of your programming assignments.

Test your program before you send it. Sending in a program that has not been adequately tested will most surely result in a poor grade. While your instructor may have supplied some example input and output data with the assignment, it is not enough to assure that your program in correct. The Grader uses its own set of input and output data which will conform to your program specification, but may provide a more rigorous test of your program than your assignment's sample data. Try a wide range of input values, and study the output carefully to be sure that it is correct.

Use the results from a submission to diagnose your problem. When you get the results back from the Grader on your last submission, it will include the input data file the Grader used to test your program, the correct output, and the output from your program. If you received a bad score, study the correct output to see if you can determine what you are doing wrong. When you try to fix the problem, test it with the Grader's input file so you can compare it to the correct output. It is important to remember that the Grader uses different input data for every submission, so make sure your program works for all possible ranges of input values, not just the assignment's sample input or the input that came back from a previous submission.

Be careful of deadlines. The clock on the Grader machine is synchronized with a timeserver, so we have great faith in its accuracy. Remember that deadlines are ill-tempered beasts. Hugging one too tightly may result in getting bitten.

Submitting From the CS Lab

If you do not have Internet access at home, you can make a submission from the Computer Science undergraduate laboratory, provided you have requested an account on the lab machines. Bring the program you want to submit on a floppy disk to McBryde Hall room 116-118. Ask for a Pentium running Windows NT (if the machine is running FreeBSD, login, type **sash halt now** and re-boot into Windows NT). The Java runtime environment will already be installed on the lab machines. Follow the instructions given at the beginning of this document to install the Grader Client software. If you are submitting from the Undergraduate Laboratory, there may be some additional steps you must follow; these directions will be available in the Lab. Don't hesitate to ask the CS 1044 GTAs for help if you have problems making your submission.

Known Bugs and Alarming Behaviors

Grader Server Crashes: In most cases, if a student submits a program that commits a runtime error (such as a divide by zero or an infinite loop), the Grader will simply kill the program, assign a score of zero, and proceed with the next submission. However, it is possible that some programs may misbehave in such a way that Windows NT prevents the Grader from killing the program; in such cases, the Grader may fail to score any submissions correctly until the offending program is killed by a human operator.

No submissions will be lost in this situation, and the Grader will automatically rescore any pending project submissions with the correct timestamp once the misbehaving program is killed. You may, however receive multiple e-mail messages in the interim, each indicating a problem with your program and a score of zero. Don't be alarmed by this if you're sure your program does run properly, but notify the Grader Administrator (`grader@cs.vt.edu`) to be sure the problem is fixed as quickly as possible.

Enrollment Issues: The Grader will indicate you are not enrolled in the class if you enter an incorrect ID number, or if your PID is not correctly recorded in the roll file the Grader uses. This may be because you selected the wrong section, entered your ID number incorrectly, or because there is an error in the roll file. If the Grader insists you are not enrolled in your section, contact your GTA or Instructor as soon as possible to resolve the problem.

E-mail Notification Problems: The rate at which the Grader is receiving submissions will determine how long it takes for an e-mail message to be sent to you; normally you should receive a message within 5 minutes. In some cases, the Grader may fail to get a connection with the campus e-mail server when it attempts to send your e-mail scoring notification. In that case, you will not receive an e-mail message, although the scoring and recording of your grade will be completed. We're working on this problem. If you don't receive an e-mail confirmation, contact your GTA and ask them to confirm the submission and send you the results.

Getting Help and Reporting Problems

Sources of Help: If you need additional help using *Submit*, see your Instructor, or your GTA, or the Consultants in the Undergraduate Lab, not necessarily in that order. If you have questions about the score your program received, bring a copy of the e-mail message sent to you by the Grader.

Bug Reports: Problems and possible bugs can be reported to the Grader Administrator by e-mail: `grader@cs.vt.edu`.

Do not send an e-mail reply to the messages from the Grader — no one will either read or respond to such messages.

The Automated Grader and the Honor Code

Each program you submit to the Automated Grader is subject to the Virginia Tech Honor Code, just as if you had given the program to a human for evaluation.

Your Instructor will specify precisely what sources of help are allowed and how much, if any, collaboration with other students is allowed. Be certain you understand and follow the rules set by your Instructor. Ignorance of those rules is not an effective defense before the Virginia Tech Honor Court.

Regardless of the rules set by your Instructor, each of the following is considered a flagrant violation of the Honor Code and will result in a formal charge:

- submitting a program written by another student as your own, individual work
- submitting a program designed to alter the operation of the Grader's scoring mechanism
- submitting a program designed to crash or otherwise damage the operation of the Grader software or the machine on which it is installed
- editing computer generated output, such as an e-mail message sent by the Grader, and presenting that altered version when raising a question relating to the scoring of your program
- attempting to access or alter files on the machine on which the Grader is installed, whether physically or via a network connection; the sole exceptions would be normal use of *Submit* and if the Grader machine is also used as an FTP repository for course-related files

This list is not intended to be comprehensive; resolve any questions you have about these policies with the Instructor of your course.

All submissions to the Automated Grader are archived. The programs submitted to the Grader are automatically analyzed for suspicious similarities. When such similarities are found, the programs involved are compared (by humans) and charges are filed with the Honor Count if the similarities warrant action.

> The Honor Code will be strictly enforced by the Instructors and GTAs who use and administer the Automated Grader. All assignments submitted shall be considered pledged graded work, unless otherwise noted. All aspects of your work will be covered by the Honor System. Honesty in your academic work will develop into professional integrity. The faculty and students of Virginia Tech will not tolerate any form of academic dishonesty.