

Western Washington University

Computer Science Department

CSCI 141 Computer Programming I Fall 2011

Laboratory Exercise 3

Objectives

1. Practice in use of the procedures from packages Ada.Text_IO and Ada.Integer_Text_IO.
2. Practice in use of selection and interaction constructs of the Ada language.
3. Practice in producing zipped tar files in Linux.

Submitting Your Work

In this lab exercise you will develop an Ada program to calculate the grades for students in a class. Save your program file (the .adb file) in the zipped tar file WnnnnnnnnLab3.tar.gz (where Wnnnnnnnn is your WWU W-number) and submit the file via the **Lab Exercise 3 Submission** item on the course web site. You must submit your program by 4:00pm on Friday, October 14, 2011.

The Exercise

In this lab exercise you will develop an Ada program to read in the scores of each student in the assessment items in a class, calculate the student's grade, collect data and report statistics on the class results.

This lab exercise is to be completed in the Linux operating system. If your computer is currently running Windows, restart it in Ubuntu Linux.

Grade Data

Instead of typing the input data at the keyboard, you can use the data provided in two files from the course web site: Class1.txt and Class2.txt. When you run your program, you can redirect the program's input to come from one of these files, instead of from the keyboard. If you call your Ada program grader.adb then, after you have compiled, bound and linked the program grader, you run it with the data by using the command

```
./grader < Class1.txt
```

The '<' symbol in the command means that the input comes from the file whose name follows it, instead of from the keyboard.

If you look at one of the data files, you will notice that each line contains 12 integer numbers. Each row represents the scores that one student obtained in the assessment items in some course. The number of rows in the data file varies from one class to another and your program cannot make any assumptions about how many rows need to be input.

Hint: The function `Ada.Text_IO.End_of_File` is true after the last number has been read from input.

The first 7 numbers in each row represent a student's scores in lab exercises. The total of the lab exercise scores is worth 15% of the student's total score in the course. The remaining 85% of the student's total score comes from the sum of the remaining 5 numbers in the row.

What your program must do

1. Read in the scores for one student.
2. Calculate the total of the 7 lab scores.
3. Calculate the student's total score for the class as
 $15 * \text{LabTotal} / 70 + \text{Item8} + \text{Item9} + \text{Item10} + \text{Item11} + \text{Item12}$
4. Determine the student's grade based on the student's total score, using the criteria:

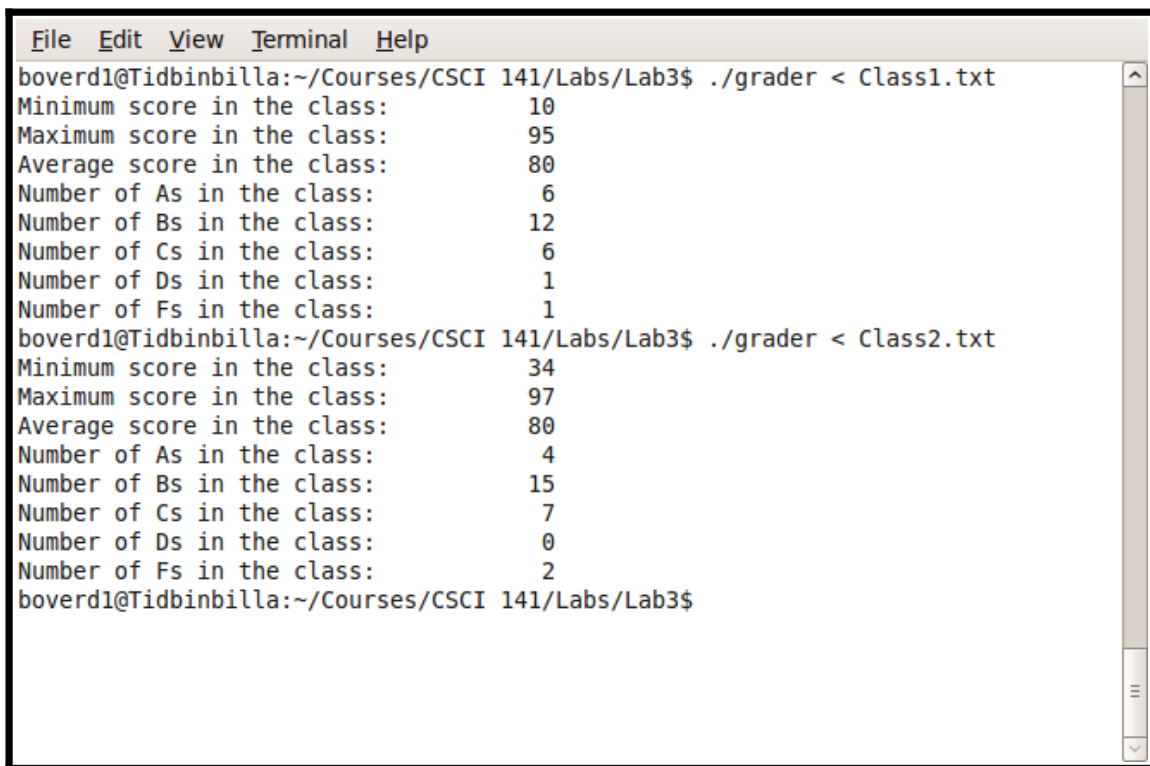
Total	Grade
$90 \leq \text{Total}$	A
$80 \leq \text{Total} < 90$	B
$70 \leq \text{Total} < 80$	C
$60 \leq \text{Total} < 70$	D
$\text{Total} < 60$	F

5. When the program has read all the data, it must output the following information:
 - The maximum total score in the class
 - The minimum total score in the class
 - The average total score in the class (integer division is sufficient for this)
 - The number of students with each grade

In order to do this, you will need to do the following:

- Have an integer variable to represent the current maximum, which is initialized to a low number (0 is low enough). When you calculate each student's total score, compare it to the current maximum. If the student's total score is larger than the maximum, it becomes the maximum.
- Have an integer variable to represent the current minimum, which is initialized to a high number (100 is high enough). When you calculate each student's total score, compare it to the current minimum. If the student's total score is smaller than the minimum, it becomes the minimum.
- Keep a count of the number of rows of data (the number of students in the class).
- Keep a running total of the student total scores.
- Keep a running total of the number of students attaining each grade.

The results for Class1.txt and Class2.txt are shown below.



```
boverd1@Tidbinbilla:~/Courses/CSCI 141/Labs/Lab3$ ./grader < Class1.txt
Minimum score in the class:      10
Maximum score in the class:      95
Average score in the class:      80
Number of As in the class:       6
Number of Bs in the class:      12
Number of Cs in the class:       6
Number of Ds in the class:       1
Number of Fs in the class:       1
boverd1@Tidbinbilla:~/Courses/CSCI 141/Labs/Lab3$ ./grader < Class2.txt
Minimum score in the class:      34
Maximum score in the class:      97
Average score in the class:      80
Number of As in the class:       4
Number of Bs in the class:      15
Number of Cs in the class:       7
Number of Ds in the class:       0
Number of Fs in the class:       2
boverd1@Tidbinbilla:~/Courses/CSCI 141/Labs/Lab3$
```

Submitting your program

1. Be sure that your program meets the coding standards listed on the next page.
2. Include your program file in a zipped tar file (see Lab 2 for instructions).

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.
2. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
3. Use comments at the start of each procedure to describe the purpose of the procedure and the purpose of each parameter to the procedure.
4. Use comments at the start of each section of the program to explain what that part of the program does.
5. Use consistent indentation:
 - The declarations within a procedure must be indented from the Procedure and begin reserved words. The body of the procedure must be indented from the begin and end reserved words. Example of procedure indentation:

```
procedure DoStuff is  
    Count : integer;  
    Valid : boolean;  
begin  
    Count := 0;  
    Valid := false;  
end DoStuff;
```

- The statements within the then-part, each elsif-part and the else-part of an if statement must be indented from the reserved words if, elsif, else and end.

```
if Count > 4 and not Valid then  
    Result := 0;  
    Valid := true;  
elsif Count > 0 then  
    result := 4;  
else  
    Valid := false;  
end if;
```

- The statements within a loop must be indented from the loop and end loop.

```
loop  
    Count := Count + 1;  
    Get (Number);  
    exit when Number < Count;  
end loop;
```