|  |  |  |
| --- | --- | --- |
| **Due:** | **Sec-A** | **Thurs. Nov. xx, 20xx** |
|  | *Sec-B* | *Thurs. Nov. xx, 20xx* |

**Assign 15 B: Selection - Practice**

CSC-110

**General Instructions:**

First create the “by hand” (sample data) part of the design. Then do any remaining parts of the design you need in order to code this program. Only the “by hand” part of design will be collected. *See “Hints” at the very end.*

NOTE: No Logical Operators (i.e. no && or ||). No Math methods (eg. math.min(), etc.), No Nesting (i.e. no testing different vars inside the if-clause or the else-clause. Only single, dual and multi-alternative selections allowed.

*Problem-1:* ***Test Summary***

Write a program named TestSummary that takes in 3 **unique** test scores **(i.e. no two have the same value)** and provides the following information: (a) test average, (b) letter grade based for that average (c) lowest test score and corresponding test number, (d) highest test score and corresponding test number and (e)points message, only if the student’s average earns them points toward a reward. The letter grade equivalencies are:

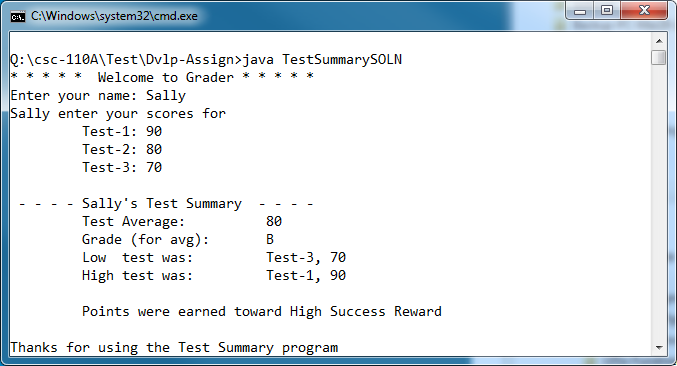
|  |  |
| --- | --- |
| Test Average | Grade |
| 90 – 100 | A |
| 80 – 90 (Excl) | B |
| 70 – 80 (Excl) | C |
| 60 – 70 (Excl) | D |
| 0 – 60 (Excl) | F |

Students qualify for reward points only if they have a test average of 80 or better.

*Specific Requirements:*

* Assume only valid test scores (0-100) are entered. Each test is 100 points.
* Use boolean var named earnedPts to handle processing (i.e. printing) of earning reward .
* Use streamlining technique if applicable.
* The default else clause must be coded if using multi-alternative processing
* Use constants to define all grade cutoffs (e.g. A\_CUTOFF)
* Show 1 place precision only when the average contains a decimal value; otherwise print only the whole number (Ex. 85 or 75.2 not: 87.0).
* Use ONLY single, dual, and multi-alternative structures (NO arrays, or loops, etc.)

*Execution Format:*



*(cont. next page)*

*Sample Execution(s):*

3 Executions: (name, test-1, test-2, test-3)

1. Sally, 90 -80 -70
2. Joe, 75 – 85 – 78
3. Fred, 87 – 63 - 73

**What to submit**:

Create a packet with:

* Execution Report: **TestSummary**
* Source Code: **TestSummary**
* Design (by hand): **TestSummary**

*Note: Verify you have correct assignment number and name AND your name.*

Be sure your packet is stapled!

**HINTs:**

1. Remember a program compares only two things at one time.
2. Before finding the low and high score “by hand” (on paper), first try it physically with cards. Take 4 or 5 different valued cards from a deck, mix them up, and find the lowest. Look at the first two cards, what do you do? Now look at the third card what do you do, look at the fourth what do you do. Did you keep any cards? If so, realize that programs use vars to keep data.
3. Take what you have learned from the deck and try that process with 3 different test scores (the ‘do by hand’).
4. Develop the program incrementally