Contestant Number:

Time:

Rank:

C++ PROGRAMMING

(335)

REGIONAL – 2019

**Production Portion:**

Job 1: Department Graph (320 points)

***TOTAL POINTS***  *(320 points)*

**Failure to adhere to any of the following rules will result in disqualification:**

1. **Contestant must hand in this test booklet and all printouts. Failure to do so will result in disqualification.**
2. **No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests or facsimile (handwritten, photocopied, or keyed) are allowed in the testing area.**
3. **Electronic devices will be monitored according to ACT standards.**

No more than ten (10) minutes orientation

No more than ninety (90) minutes testing time

No more than ten (10) minutes wrap-up

Property of Business Professionals of America.

May be reproduced only for use in the Business Professionals of America

*Workplace Skills Assessment Program* competition.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on any work you submit for grading.

Your application will be graded on the following criteria:

**Solution and Project**

Project was found on the flash drive \_\_\_\_ 10 points

Project is named following the naming convention \_\_\_\_ 10 points

**Program Execution**

Program runs \_\_\_\_ 20 points

**If program does not execute, then remaining items in this section are not scored.**

The program gracefully handles any data errors \_\_\_\_ 20 points

The program shows a department menu to choose from \_\_\_\_ 10 points

The program displays a prompt for collecting student sizes from the user

which includes which class is currently be collected. \_\_\_\_ 10 points

The program displays an error message for bad user data \_\_\_\_ 10 points

The program displays both types of graphs \_\_\_\_ 20 points

The program displays prompts for user control \_\_\_\_ 20 points

The program shows all data on clean console screens \_\_\_\_ 10 points

The program allows the user to rerun the program without closing

and restarting the current copy. \_\_\_\_ 5 points

**Source Code Review**

Contestant ID is commented at the top of the program \_\_\_\_ 5 points

Code is commented at the top, for each method and as needed \_\_\_\_ 15 points

Code uses reasonable and consistent variable naming conventions \_\_\_\_ 15 points

The program reads the class sizes into the correct data structure \_\_\_\_ 10 points

The program gracefully handles remaining error checking \_\_\_\_ 20 points

A method called “menu\_collect” is implemented \_\_\_\_ 30 points

A method called “horizontal\_graph” is implemented \_\_\_\_ 30 points

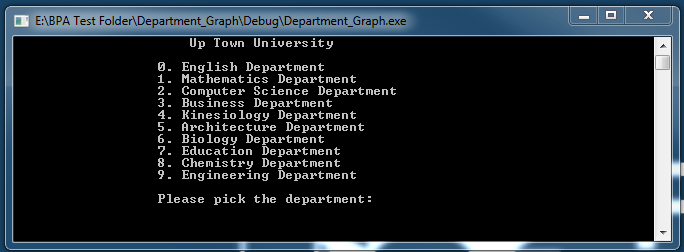
A method called “vertical\_graph” is implemented \_\_\_\_ 40 points

The program uses the bool data type in a logical manor \_\_\_\_ 10 points

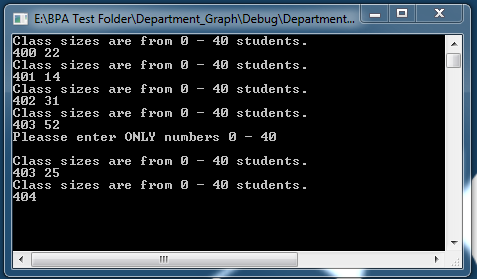
**Total** \_\_\_\_/320 points

**Department Graph**

The local university – Up-Town University is looking for a program to display in graph form the number of students in each class of a given department. The university wants a program that will take a departments ID# ( a single digit number 0 – 9), and then ask for the number of students found in that departments 10 classes. The classes are always the department’s number plus 00 – 09. A class can have a student size of 0 – 40. Let the user pick from a menu the department they wish to see (Fig. #1).

 Fig. #1

Collecting a department’s classes should look something like this:

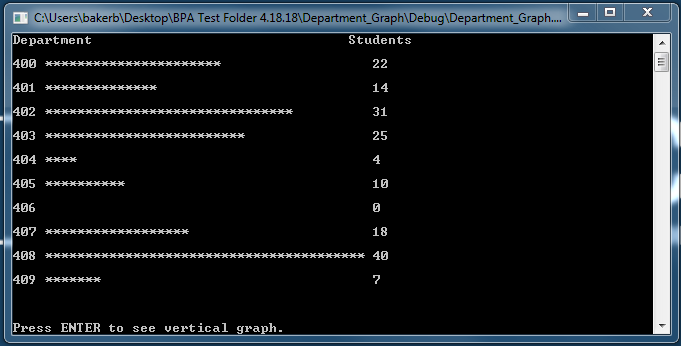
 Fig. #2

**PLEASE NOTE: The user entered an incorrect value and was reminded for correct valid answers only.**

You must ask for all the student numbers first then display the class size using the following two forms:

**Horizontal Graph**

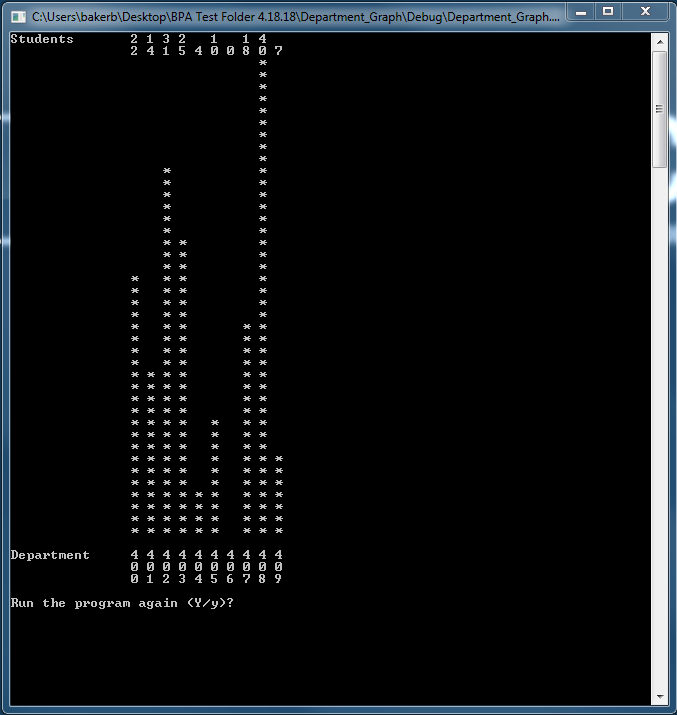
Take the data entered for the department’s classes and display them in a horizontal graph. The output might look like this:

 Fig.#3

Make sure to hold the graph on the screen until the user is ready to continue.

**Vertical Graph**

Now display the same data in a vertical graph. The output might look like this:



Remember to ask the user if they would like to run the program again. Each new screen should be cleared of old before new data is displayed. When saving the program remember to call it Depart\_Graph.