Using your textbook, labs and the internet complete the following test. All code submitted must be your original work.

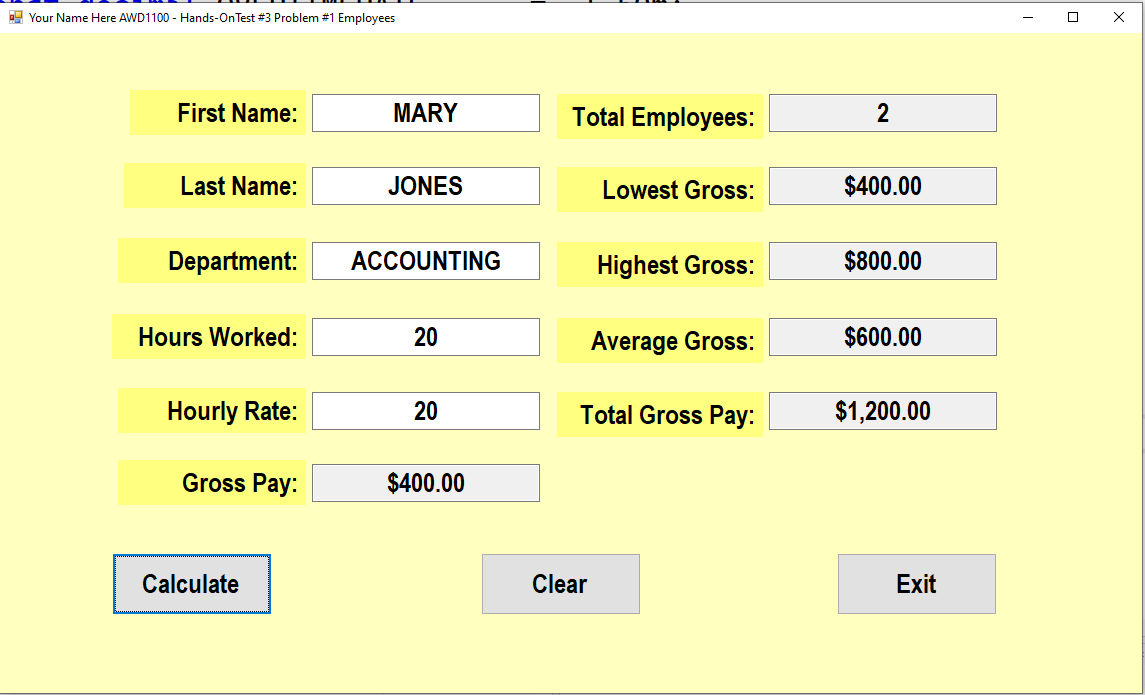
Please create the projects under the **same solution** used for the previous test and upload Completed work to **existing** GitHub (10 points).

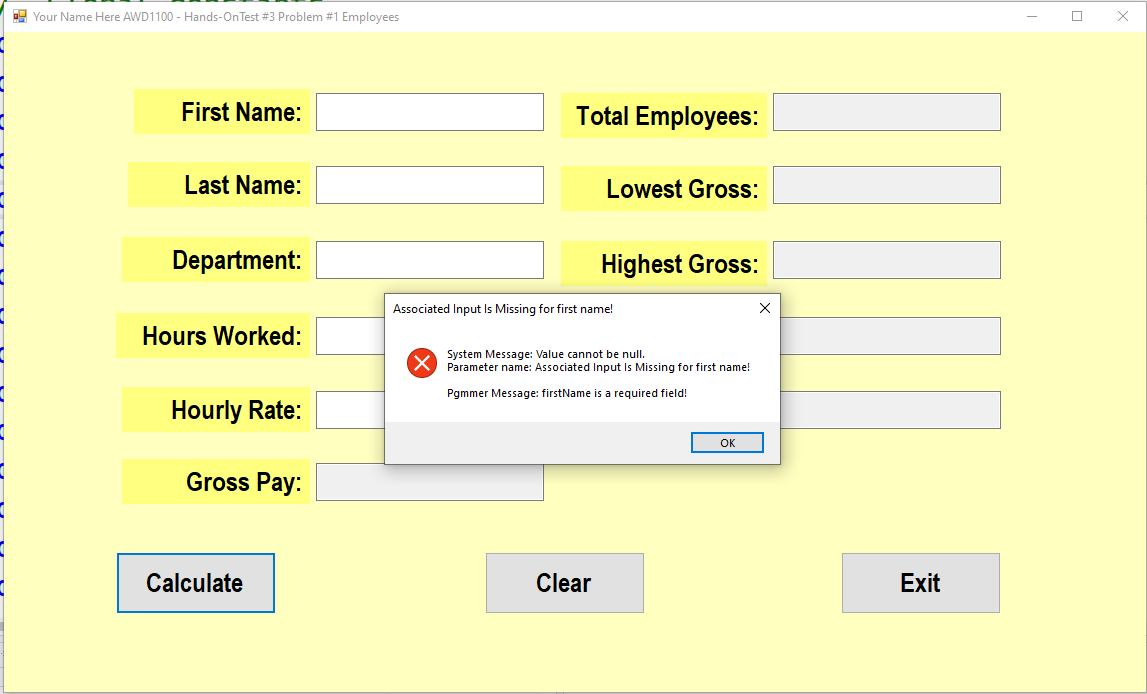
1. [90 pts] Write a GUI-based program that create a simple payroll system. Please see the next page for program run examples.

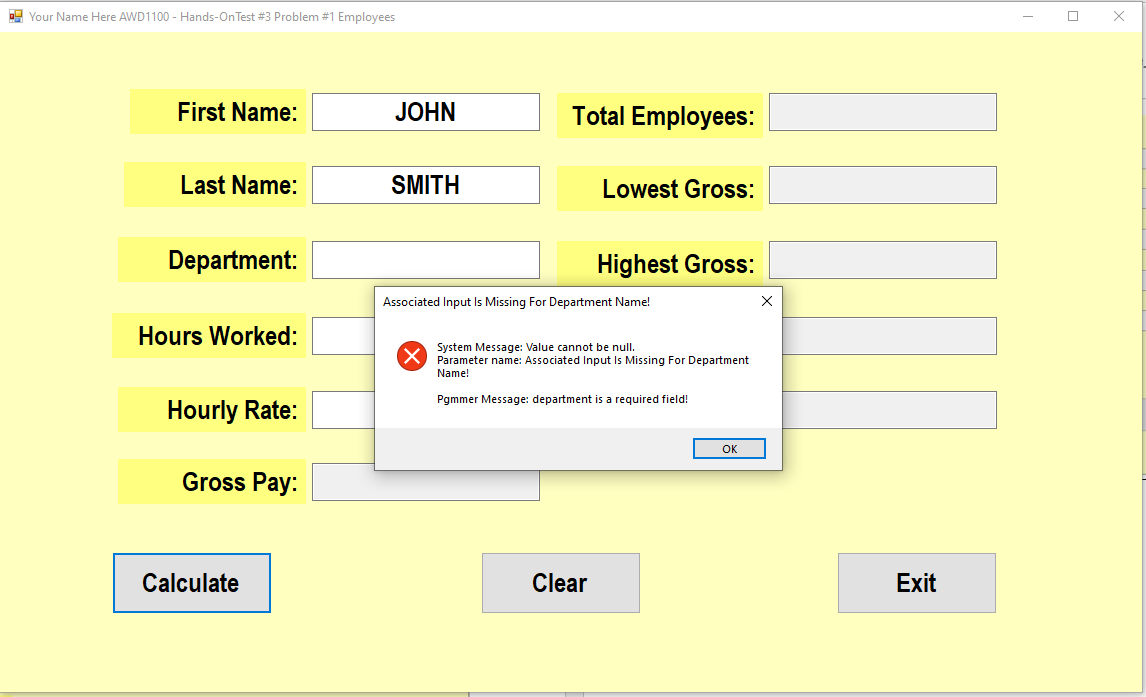
* **The program must use constants** for department (possible values are INFOTECHNOLOGY with value "INFO TECHNOLOGY", WEBDEVELOPMENT with value "WEB DEVELOPMENT", PROGRAMMING with value "PROGRAMMING", ACCOUNTING with value "ACCOUNTING", and SHIPPING with value SHIPPING"), minimum hours worked (0), maximum hours worked (84), minimum hourly rate (0), maximum hourly rate (99.99), maximum hours worked with no overtime (40), and overtime rate (1.5). I also used constants for my titles. See Pages 3 and 4 for examples.
* **You must use try/catch exception handling for the following**: a) first name and last name cannot be empty. b) department cannot be empty. c) department must be 1 of the 5 constants listed above. d) hours worked and hourly rate cannot be empty. e) hours worked and hourly rate must be numeric. f) hours worked and hourly rate must be with range of constants listed above. Correct program runs are shown on the next page. Error program runs are shown on Pages 3 – 4. Try to use the generic Murach error handlers Page 213.

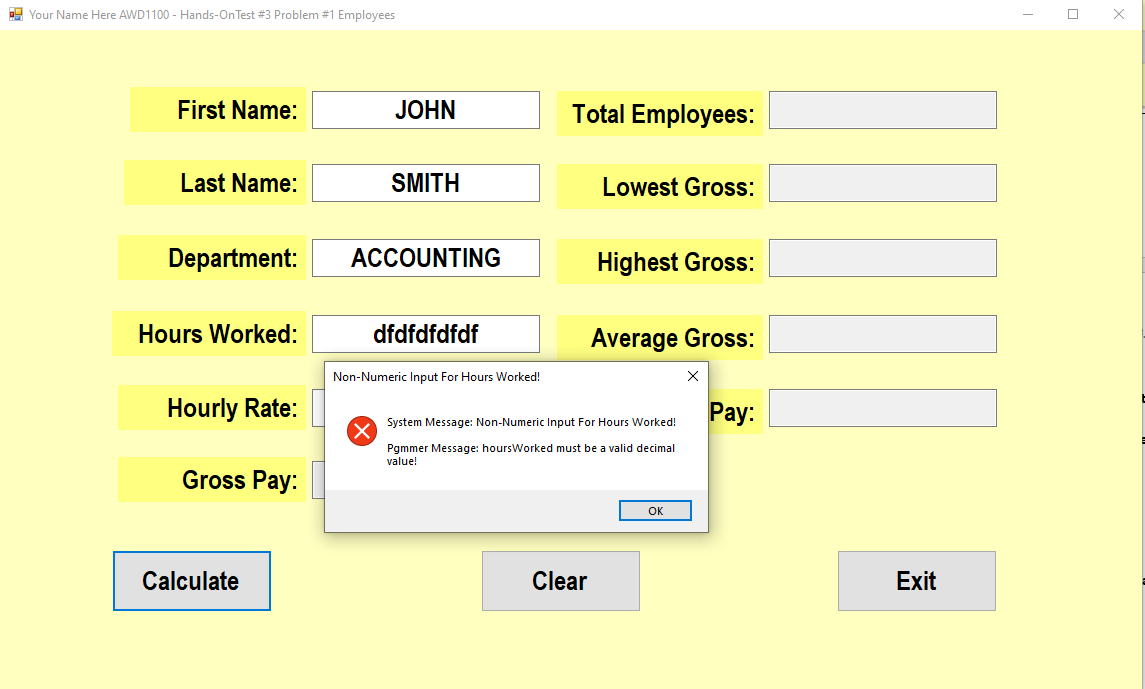
Graphical user interface, table

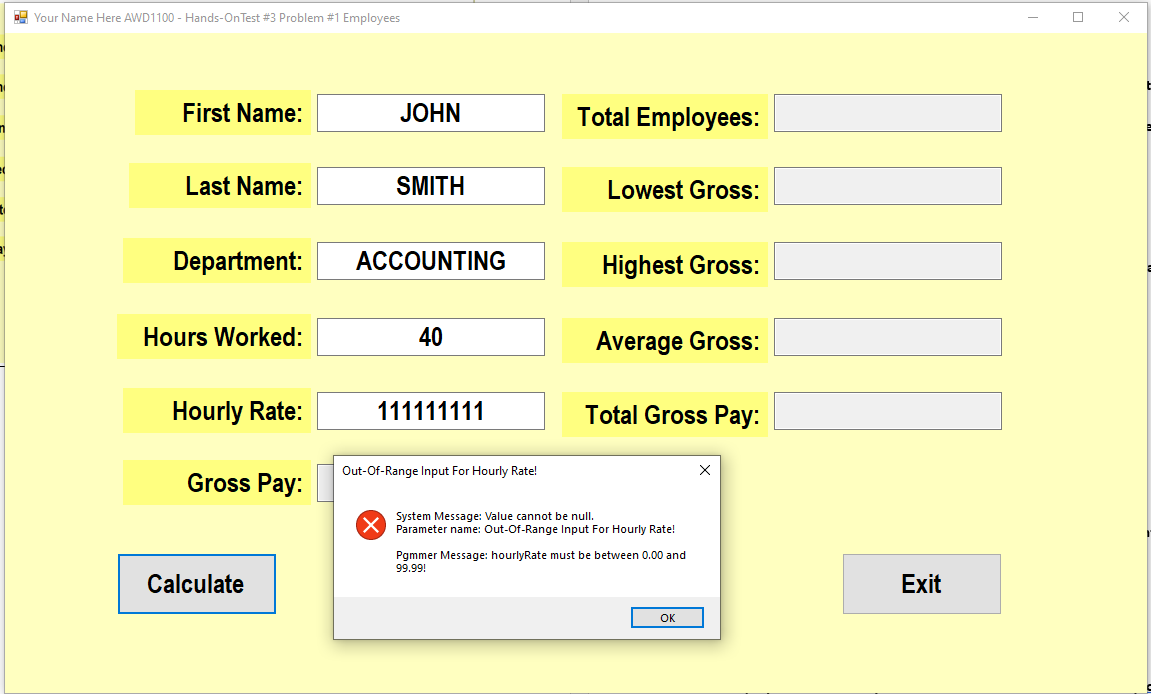
Description automatically generated

****

****

****

****



Grading:

As noted, project must be saved to your **existing** HOTs solution & uploaded to your **existing** GitHub HOTs repo. 10 points total.

Make sure for the project: 1 point each, 8 points total:

a) Has AcceptButton (btnCalculate) and CancelButton (btnClear) set correctly.

b) Has the tab order set *naturally*.

c) Has the windows Start position set to Center Screen.

d) Uses self-documenting variable names and method names.

e) Contains a project prologue at the beginning. Copy from Page 1 of this test as necessary.

f) Contains additional internal comments as necessary.

g) Change the form name for the project to frmEmployees.

h) Change the form text for the project to what is shown in the examples, replacing “Your Name Here” with your actual name.

Grading (continued):

Program uses modularization. Buttons act like drivers, rather than containing excessive code – 10 points

Program correctly uses exception handling, including try/catch statements. – 15 points

Read-only textboxes are used where shown, as shown – 2 points

Program contains (minimally) all constants shown on page 1, each constant in UPPERCASE format – 5 points

Program accepts first name, last name, department, hours worked, and hourly rate and handles any/all bad inputs as shown – 10 points

Program calculates and displays gross pay correctly (including overtime if worked) – 15 points

Program displays all output in the desired format (ALL read-only textboxes should be in currency format. All non-numeric fields should be in UPPERCASE as shown in examples) – 5 points

Entire program (including any/all conditional statements) runs without syntax errors or warnings – 5 points

Program correctly keeps track of all totals and running totals and displays all in format shown – 15 points