

GNG1106 Fall 2016 - Assignment 1

Available: September 5

Due: September 25, midnight

Instructions

This assignment is to be done INDIVIDUALLY. Follow the instructions in the GNGBlackboard document that describes how to submit assignments through the Blackboard. The following are specific instructions for this assignment:

- You will need to submit your assignment electronically to Blackboard. Submit the following
 - An assignment report in a PDF file (this allows you to use your favorite editor to create the PDF file). For question 1, insert the programming models for parts (a) and (b) filled in as per the question instructions. You may fill in the programming model using drawing features of your editor or by hand on paper which is then scanned and inserted into your document. For Questions 2 and 3, insert in your assignment report the source code (take care in its appearance), and capture the output from running the program for all test cases. Also submit your source code files for questions 2 and 3.
 - Place all your files (PDF file and C source code files) in a directory A1_xxxxxxx where xxxxxxxx is your student number.
 - Zip your PDF document and the C source files in a zip file with the name A1_xxxxxx.zip where xxxxxx is your student number and submit the zip file before the assignment deadline.
 - The questions are provided in both PDF and Word files. You may use the Word file to enter your answers in the document. An rtf file is also provided so that you may edit the file with a word processor other than Word.
 - It is NOT permitted to use instructions such as branches and loops that have not yet been covered in the lectures.
- Do start the assignment soon and do **not** wait until the last minute. You will be more efficient with a number of smaller efforts over a few weeks before the deadline than one large effort just before the deadline.

Marking Scheme (total 35 marks)

- Question 1: 10 marks
- Question 2: 10 marks
- Question 3: 15 marks

Question 1 (10 marks)

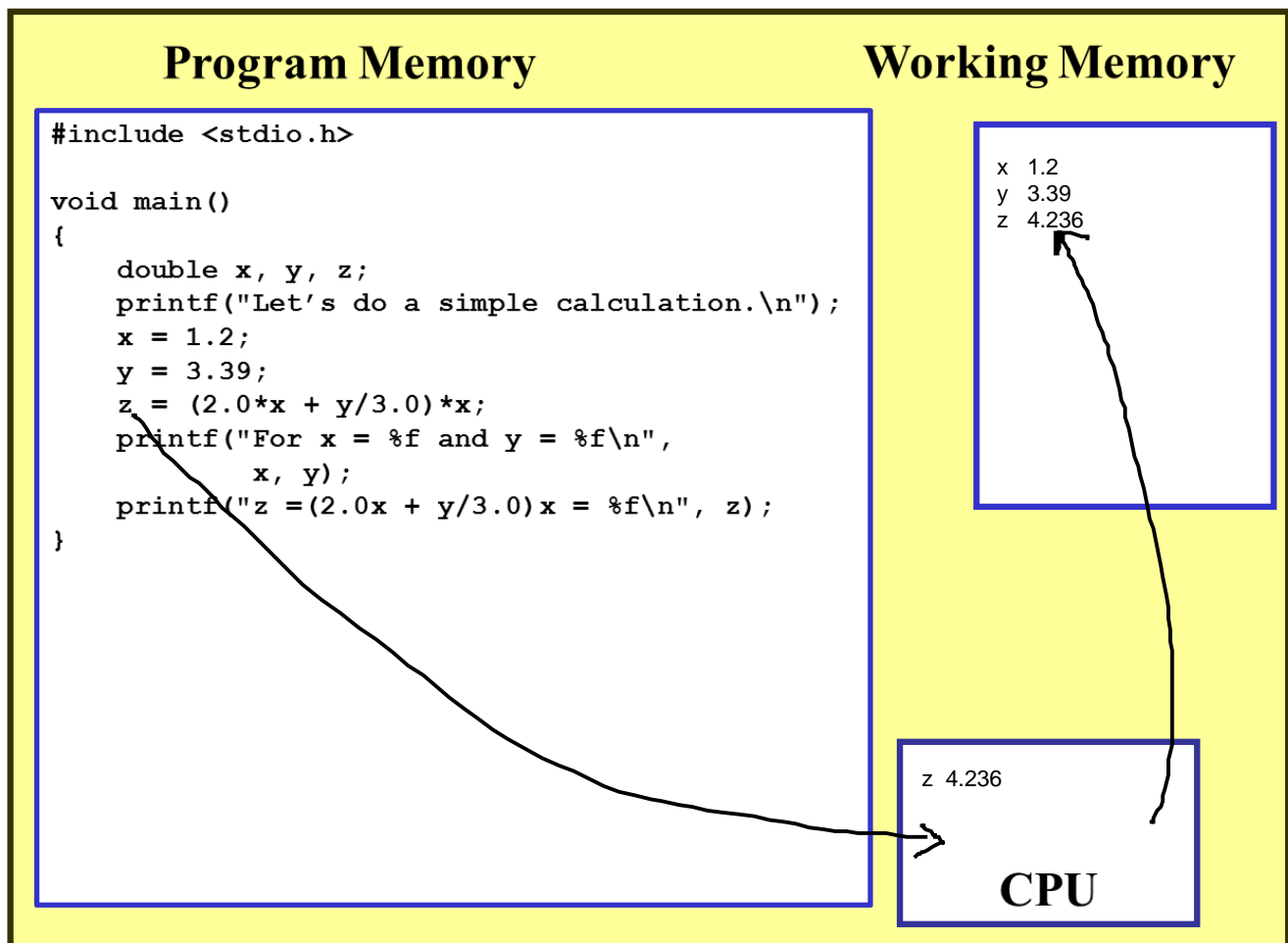
- a) The following programming model contains the indicated C program in its program memory. You will be showing how the working memory is used during the execution of this program and how the CPU evaluates the arithmetic expression that assigns a value to the variable **z**.

Show the variables created in the working memory and how their values change during the execution of the program.

- Show the values are assigned to the variables. Be sure to show all values that are assigned and replaced. Record successive assignments to variables/parameters as follows:

Variable ~~7~~, ~~2~~, ~~6~~, ~~4~~, 10

- For the operation, show how contents from the working memory are moved to the CPU to calculate the value assigned to variable **z**. Show the results of **all** operations carried out in the CPU, that is, one line per operation.



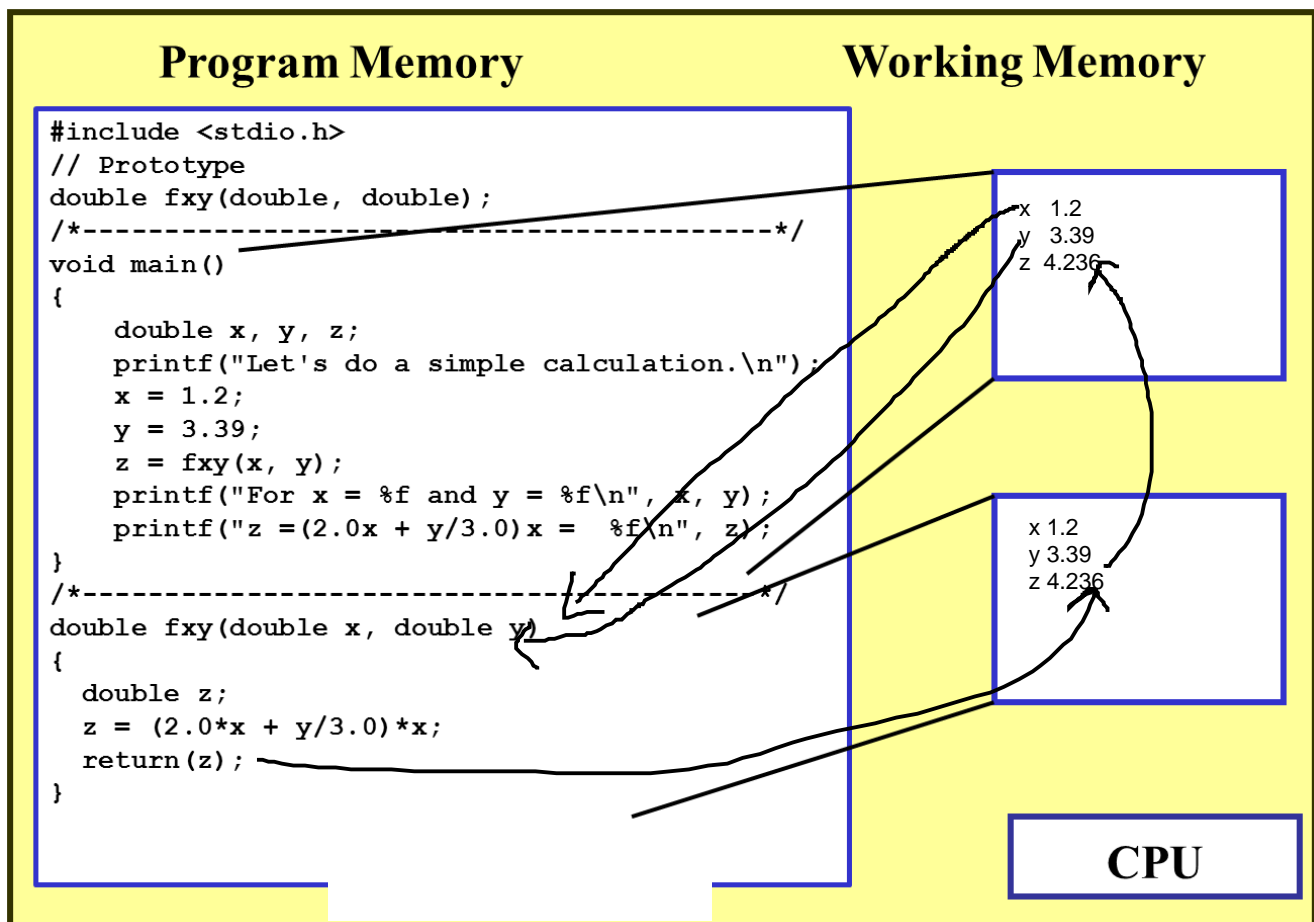
- b) The following programming model contains in its program memory the indicated C program composed of 2 functions. You will be showing how the working memory is used during the execution of the two functions. Each piece of working memory is associated to a function using a pair of lines. (Note: the first pair of lines associates the piece of working memory allocated to the function **main** and the second pair of lines associates the piece allocated to the function **fx**y).

Show how the variables (and parameters) are created in each piece of working memory during the execution of the functions.. It is **not** necessary to show how the operations are carried out in the CPU as in the case of part (a).

- Show the values are assigned to the variables. Be sure to show all values that are assigned and replaced. Record successive assignments to variables/parameters as follows:

Variable ~~7~~, ~~2~~, ~~6~~, ~~4~~, 10

- Using arrows show how values are copied between the working memory allocated to the function **main** and the working memory allocated to the function **fx**y.



Question 2 (10 marks)

The following equations are well known relationships between the constant acceleration a , velocity v , and displacement x of an object that travels in a single direction (x).

$$v = v_0 + at$$

$$x = v_0 t + \frac{1}{2} at^2$$

where a is a constant acceleration in m/s^2 ,
 v is the velocity in m/s ,
 x is the displacement in m ,
 v_0 is the initial velocity at time $t = 0$.

Develop a program that requests from the user the initial velocity at time 0, v_0 , and the acceleration, a , of an object travelling in a straight line and a time t (greater than 0). From these values, the program shall calculate the displacement, x , of the object after time t .

Test your program using the cases shown in the following table.

Time t (seconds)	Initial Velocity v_0 (m/s)	Acceleration (m/s)	Velocity (m/s)	Displacement x (m)
10	1	0	1.00	10.00
0.5	0	250	125.00	31.25
5.2	10.2	0.5	12.80	59.80
120	60	1.2	204.00	15,840.00
0	60	1.2	60.00	0.00

Guidelines:

Logic/Strategies

- In the **main** function
 - Consider using the following local variables.
 - **t**: for storing time given by user.
 - **v0**: for storing the initial velocity given by the user
 - **a**: for storing the acceleration given by the user
 - **x**: to store the calculated distance.
 - Prompt the user for the values of time, initial velocity and acceleration. Use appropriate prompt messages. Read the values from the keyboard and assign them respectively to the variables **t**, **v0** and **a**.
 - Call a function, say **calculateDistance**, to calculate the value of distance travelled after time **t**. Store the result returned by the function in the variable **x**. You will need to define this function in your program.
 - Display the results with a message of the following form:
“After time $t = 10.2$ seconds, the displacement $x = 59.80$ m”
- For the function **calculateDistance**
 - Consider using the following parameters
 - **t**: gives the time the object travelled.
 - **v0**: gives the initial velocity of the object
 - **a**: gives the object's acceleration.

- Consider using the following local variables
 - **x**: to store the calculated distance (note that this variable will contain the value returned).
- The function calculates the distance travelled (see the equation given at the beginning of the question), stores it in the variable **x** and returns the value stored in **x**.

Test the program using the values provided in the table above. In your assignment report, give the output of your program for all test cases.

The answer to this question should provide:

- 1) The source code to your program (also insert the source code into the assignment report).
- 2) The output showing the results of all the test cases; insert the output into the assignment report. The following is an example of the output for the first test case.

```
d:\UofO\Courses\CurrentCourses\GNG1106\Fall2016\Assignments\A1\Assignment01_Q2.exe
Please enter a time (sec.): 10
Please enter initial velocity (m/s): 1
Please enter acceleration (m/s^2): 0

Results:
After time 10.00 s, The distance x=10.00 (m)
Process returned 55 (0x37) execution time : 7.862 s
Press any key to continue.
```

Question 3 (15 marks)

The ideal gas law, first stated by Émile Clapeyron in 1834 provides a simple relationship between the pressure, temperature and mass of a gas.

$$PV = \frac{m}{M} RT$$

where P is the gas pressure in units of kPa (kilo pascals),

V is the gas volume in cubic meters (m^3),

T is the temperature in degrees Kelvin,

M is the molecular weight of the gas in kg/kmole (for air, $M = 28.97$ kg/kmole),

m is the weight of the gas in kg,

R is the ideal gas constant with the value $8.314 \text{ kPa m}^3/(\text{kmole K})$.

Using the ideal gas law, develop a program that determines the volume of a gas assuming it is under a pressure of 1 atmosphere ($1 \text{ atm} = 101.325 \text{ kPascals}$) and that the mass, molecular weight and temperature (in degrees Celsius = Kelvin – 273.15) are obtained from the user. (Hint: Be careful of the units.)

Complete this question as follows:

- 1) First develop a set of test cases that include the following gases (Excel is practical spreadsheet software that allows you to create test cases). Be sure that test cases cover wide ranges for the input data (small and large mass, low and high temperature, etc.) and that the selected substance is in the gaseous state (i.e. use the gas boiling point to select an appropriate temperature). Provide at least 5 test cases (one case for each substance in the table below). For example, 0.1 kg of argon at -50 Celsius and at a pressure of 1 atm (101.325 kPascals) has a volume of 0.045629 m^3 . The following are examples of values for various gases.

Gas	Molecular Weight (kg/kmole)	Boiling Point (degrees Celsius)
Argon	39.948	-185,8
Benzene	78.114	80.4
Hydrogen	2.016	-253
Nitrogen	28.0134	-196
R-114 (refrigerant)	170.93	3.59

- 2) Develop your program using the GNG1106 C template (GNG1106template.c), that is, your program will contain a `main` function and a function that calculates the volume of the gas. The `main` function contains instructions to get data from the user, calls the function to get the volume of the gas, and displays the results to the user.
- 3) Include in your assignment report, a table of your test cases, your source code, and the captures output for all your test cases. Also submit your source code file.