INTRODUCTION TO PROGRAMMING
BERKELEY CITY COLLEGE
FALL 2019
LAB 1

It is very important that you attend lectures and labs. Many of these problems will be discussed in class.

Use good programming practices. Display messages so that the results are easily understandable.

1. **Nametag**
   At a certain conference, each delegate has to wear a tag showing his name and the organization they represent. The name will be written on the tags by hand. Write a program to print the following blank nametag:

```
##############################################################################
###                          ANNUAL CONFERENCE                             ###
##############################################################################
### NAME:                                                                  ###
###                                                                        ###
##############################################################################
### ORGANIZATION:                                                          ###
###                                                                        ###
##############################################################################
```

2. **Distance between two points**
   The distance between two points (x1, y1) and (x2, y2) on a Cartesian coordinate plane is given by the equation

$$D = \text{sqrt}\left((x1 - x2)^2 + (y1 - y2)^2 \right)$$

   Write a program to calculate the distance between any two points (x1, y1) and (x2, y2) specified by the user. Use good programming practices in your program. Use the program to calculate the distance between (2, 3) and (8, -5).

3. **Karel (version 0.1)**

Write a program that accepts the following commands from the user:

```
move();
turnLeft();
quit
```

If the user enters any other command, ask them for good input.

Create three variables:

```
int karelRow
int karelColumn
string karelFacing
```

Initialize both karelRow and karelColumn to 0. Initialize karelFacing to "east".
Each time the user enters a command (other than quit) move Karel based on that command and the direction Karel is currently facing.

If Karel is facing east when the user enters move();, add 1 to karelColumn.
If Karel is facing west when the user enters move(); subtract 1 from karelColumn.
If Karel is facing north when the user enters move();, subtract 1 from karelRow
if Karel is facing south when the user enters move();, add 1 to karelRow

Each time the user enters a command, print Karel's location and the direction Karel is currently facing.

Assume that Karel is on an 8x8 world. If Karel tries to move outside the bounds of this world (i.e. if Karel tries to move to row or column -1 or row or column 8), print a message saying that Karel has crashed and then exit the program.

When the user enters the command quit, print out Karel's final location and facing and thank the user for not crashing Karel.

4. **Circle**

Write a program that prompts the user for a radius *r*, and then prints

- The area and circumference of a circle with that radius.
- The volume and surface area of a sphere with that radius.

Display the radius value and the results in a clear format.

Here are the formulae (assume $\pi = 3.14$):

For a circle,

circumference = $2 \Pi r$
area = Πr^2

For a sphere,

volume = $4 \Pi r^3 / 3$

surface area = $4 \Pi r^2$

Test the results of your program for a circle and sphere with a radius of 7.5.

5. **Change Calculator**
   Write a program that directs a cashier how to give change. The program has two inputs: the amount due and the amount received from the customer. Display the dollars, quarters, dimes, nickels, and pennies that the customer should receive.

6. **Tax Calculator**
   Write a program to prompt for the income, then calculate and display the initial amount, the amount after deductions, and the deducted amount. Use good programming practices. Display messages so that the results are understandable.

   Assume individuals are taxed at an incremental rate as given below:

Incremental Income	**Tax Rate**
0 to $ 30, 000.00	0%
$30,000.01 to $50,000	10%
$50,000.01 to $100,000	20%
$100,000.01 to $200,000	30%
$200,000.01 to $250,000	35%
>250.000.01	40%

   Note that the above ranges are not total income but additional income. For example, if one earns $125,000 per year, then there is no tax on the first $30,000. The next $20,000 is taxed at a rate of 10 percent, the next $50,000 is taxed at a rate of 20 percent, and the remaining $25,000 is taxed at a rate of 30 percent.

7. **Grades**
   Write a program that prompts the user to enter the points earned for a course and determines the grade based on the following rules.

Points earned	**Grade**	**Message String**
0 <= x < 60	F	Try Harder Next Time
60 <= x < 70	D	Try Harder Next Time
70 <= x < 80	C	Good
80 <= x < 90	B	Very Good
90 <= x	A	Excellent

   The program should display the grade and the message string for on the grade. Use ‘if else’ statements to determine the grade and the switch statement to print the message string.

8. **Multiplication by addition**
   Write a program that executes the multiplication of two numbers entered by the user using only the addition operation.

   For example, if the user enters 3 and 4, the program computes the multiplication of these numbers by adding the number 4 three times (i.e., 3 * 4 = 4 + 4 + 4).

   Solve this problem in the most efficient way.

9. **Reversing a number**
   Write a program that displays the reversed number. For example, if the user enters 68743, the program should display 34786

10. Describe the *algorithm* you use for looking up a person's telephone number in the phone book. The input is person's name; the output is the corresponding phone number. No program required.

11. Describe an *algorithm* that takes two inputs: a list of *m* numbers and an integer *n* (n <= m), and returns the *n*th smallest element in the list. No program required.

12. Make a brief list of the programming errors you have encountered so far and what you should do to avoid them.