COP 3223 Assignment #5

Program A: Pizza Ordering (ordering.c):

Your friend is looking to open up a fast-food pizza restaurant. He's got everything ready so far, but last minute, he realized that no one working there knows how to calculate order totals. He's asked you to write a program that prompts users to enter the menu options they want (as many as they want), then outputs the total order cost with tax. Refer to the sample below for the menu item names and prices. An input of 0 (zero) should be the case for ending the loop and outputting the total with tax.

To simplify the process, your program should store sales tax using a constant:

```
#define SALES_TAX .06
```

Your program should have (and use) the following functions (written by you):

```
// Pre-condition: option will be a valid option number for the menu.
// Post-condition: Returns the cost of this item.
double getPrice(int option);

// Pre-condition: option will be a valid option number for the menu.
// Post-condition: Outputs the name of this item.
void printOptionName(int option);

// Pre-condition: None
// Post-condition: Outputs menu options with their numbers and costs.
void printMenu();
```

Note: you must use these functions to perform these tasks. The names of the items can only be written in the code for `printOptionName`, and the prices for the items can only be written in the code for `getPrice`. The functions `printMenu()` and `main()`, as well as any others, must call those first two functions to get these values, they cannot have the menu values coded within them.

Sample Run (user input in bold and italics):

```
0. (Complete order)
1. Small Pizza ****** 5.99
2. Medium Pizza ****** 6.99
3. Large Pizza ****** 7.99
4. Wings ****** 10.50
5. Drink ****** 3.50
```
2
```
Added Medium Pizza
0. (Complete order)
1. Small Pizza ****** 5.99
2. Medium Pizza ****** 6.99
3. Large Pizza ****** 7.99
4. Wings ****** 10.50
5. Drink ****** 3.50
```
5
```
Added Drink
0. (Complete order)
1. Small Pizza ****** 5.99
2. Medium Pizza ****** 6.99
3. Large Pizza ****** 7.99
4. Wings ****** 10.50
5. Drink ****** 3.50
```
0

```
Total: $11.12
```

Program B: Twin Prime Tester (twinprime.c):

Travis is easily amused by various math patterns. This week, his favorite pattern is what is known as twin primes. Twin primes are a pair of prime numbers that are exactly two apart. For example, 11 and 13 are twin primes, since they are both prime and are exactly 2 apart. (Remember, a prime number is a number not evenly divisible by anything except 1 and itself.)

To entertain Travis for the week, you would like to write a program that will determine whether a given number is a twin prime with the prime number that comes next after it. Additionally, Travis might try to mess with you by entering a number that is not prime, so the first number you use in the pair should be the next highest prime number if the entered one is not prime. (Note the number entered will always be greater than 1.)

Your program should have (and use) the following functions (written by you):

```
// Pre-condition: value will be an integer greater than 1.
// Post-condition: Returns true (1) if value is prime, or
//    false (0) otherwise.
int isPrime(int value);

// Pre-condition: current will be a positive integer greater than 1.
// Post-condition: Returns the next prime number greater than or
//    equal to current.
int nextPrime(int current);
```

Sample Run (User input in bold and italics):

5

```
5 and 7 are twin primes.
```

Sample Run (User input in bold and italics):

10

```
11 and 13 are twin primes.
```

Sample Run (User input in bold and italics):

36

```
37 and 41 are not twin primes.
```

Sample Run (User input in bold and italics):

2

```
2 and 3 are not twin primes.
```

Program C: Power Comparison (powcompare.c):

Stephen is practicing exponentiation. So he's decided to practice comparing one number raised to a power with another number raised to another power. But sometimes he messes up, and they aren't equal. When this happens, give him a chance to re-enter the second set of numbers until they do equal.

The first example is $2^4 = 4^2 = 16$.

Your program should have (and use) the following function (written by you):

```
// Pre-condition: base and pow will be nonnegative integers.
// Post-condition: Returns base raised to the power of pow.
int power(int base, int pow);
```

Sample Run (User input in bold and italics):

```
Enter baseA and powerA
2 4
Enter baseB and powerB
4 2
2^4 = 4^2 = 16
```

Sample Run (User input in bold and italics):

```
Enter baseA and powerA
5 6
Enter baseB and powerB
10 3
Not equal.
Enter baseB and powerB
25 3
5^6 = 25^3 = 15625
```

Sample Run (User input in bold and italics):

```
Enter baseA and powerA
11 2
Enter baseB and powerB
121 1
11^2 = 121^1 = 121
```

Deliverables:

Please submit three separate .c files for your solutions to these problems via WebCourses by the designated due date:

Program A: **ordering.c**
Program B: **twinprime.c**
Program C: **powcompare.c**