CPSC 4970 – C++ Programming
Homework 2

Your solution to each problem should include the C++ source code and the screenshot of a test run. A solution without a test run will have points deducted.

1. (**Shipping Charges**) The Fast Freight Shipping Company charges the following rates:

Weight of Package (in Kilograms)	Rate per 500 Miles Shipped
2 Kg or less	$1.10
Over 2 Kg but not more than 6 Kg	$2.20
Over 6 Kg but not more than 10 Kg	$3.70
Over 10 Kg but not more than 20 Kg	$4.80

Write a program that asks for the weight of the package and the distance it is to be shipped, and then displays the charges.

Input Validation*: Do not accept values of 0 or less for the weight of the package. Do not accept weight of more than 20 Kg (this is the maximum weight the company will ship). Do not accept distances of less than 10 miles or more than 3,000 miles. These are the company's minimum and maximum shipping distances.*

2. (**Geometry Calculator**) Write a program using the ***switch*** **statement** to display the following menu:

```
Geometry Calculator
1. Calculate the Area of a Circle
2. Calculate the Area of a Rectangle
3. Calculate the Area of a Triangle
4. Quit
Enter your choice (1-4):
```

If the user enters 1, the program should ask for the radius of the circle and then displays its area. If the user enters 2, the program should ask for the length and the width of the rectangle and then display the rectangle's area. If the user enters 3 the program should ask for the length of the triangle's base and its height, and then display its area. If the user enters 4, the program should end.

Input Validation: Display an error message if the user enters a number outside the range of 1 through 4 when selecting an item from the menu.

3. (**Game 23**) The game of "23" is a two-player game that begins with a pile of 23 toothpicks. Players take turns, withdrawing either 1, 2, or 3 toothpicks at a time. The player who withdraws the last toothpick loses the game. Write a human vs. computer program that plays "23". The human should always move first. When it is the computer's turn, it should play according to the following rules:

- If there are more than 4 toothpicks left, then the computer should withdraw 4-X toothpicks, where X is the number of toothpicks the human withdrew on the previous run.
- If there are 2 to 4 toothpicks left, then the computer should withdraw enough toothpicks to leave 1.

- If there is 1 toothpick left, then the computer has to take it and loses.

When the human player enters the number of toothpicks to withdraw, the program should perform input validation. Make sure that the entered number is between 1 and 3 and that the player is not trying to withdraw more toothpicks than exist in the pile.

4. (**Estimates of Pi**) An approximate value of pi can be calculated using the series given below:

$$pi = 4 [1 – 1/3 + 1/5 – 1/7 + 1/9 ... + ((-1)^n)/(2n + 1)]$$

Write a program to calculate the approximate value of pi using this series. The program takes an input *n* that determines the number of terms in the approximation of the value of pi and outputs the approximation.

5. **(Pattern Display)** Write a program that uses a loop to display the pattern below.

```
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
*******************
```

6. (**Random Number Guessing Game**) Write a program that generates a random number and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." The program should use a loop that repeats until the user correctly guesses the random number.

7. (**Overloaded Functions**) Write an overloaded function **max** that takes either two or three parameters of type double and returns the largest of them.

8. (**Star Search - Call-by-reference Functions**) A particular talent competition has five judges, each of whom awards a score between 0 and 10 to each performer. Fractional scores, such as 8.3, are allowed. A performer's final score is determined by dropping the highest and lowest score received, then averaging the three remaining scores. Write a program that uses this method to calculate a contestant's score. It should include the following function:

- `void getJudgeData()` should ask the user for a judge's score, store it in a **reference parameter** variable, and validate it (Do not accept a judge score lower than 0 or higher than 10). This function should be called by `main` once for each of the five judges.

- `void calcScore()` should calculate and display the average of the three scores that remain after dropping the highest and lowest scores the performer received. This function should be called just once by `main` and should be passed the five scores.

The last two functions, described below, should be called by `calcScore`, which uses the returned information to determine which of the scores to drop.

- `double findLowest()` should find and return the lowest of the five scores passed to it.
- `double findHighest()` should find and return the highest of the five scores passed to it.

9. (**Oven Keypad - Boolean Functions**) The keypad on your oven is used to enter the desired baking temperature and is arranged like the digits on a phone:

1	2	3
4	5	6
7	8	9
	0	

Unfortunately, the circuitry is damaged and the digits in the leftmost column no longer function. In other words, the digits 1, 4, 7 do not work. If a recipe calls for a temperature that cannot be entered, then you would like to substitute a temperature that can be entered. Write a program that inputs a desired temperature. The temperature must be between 0 and 999 degrees. If the desired temperature does not contain 1, 4, or 7, then output the desired temperature. Otherwise, compute the next largest and the next smallest temperature that does not contain 1, 4, or 7 and output both.

For example, if the desired temperature is 450, then the program should output 399 and 500. Similarly, if the desired temperature is 375, then the program should output 380 and 369.

Write a **function** named `containsDigit` to determine if a number contains a particular digit. The header should look like:

```
bool containsDigit (int number, int digit);
```

If *number* contains *digit*, then the function should return *true*. Otherwise, the function should return *false*. Your program should use this function to find the closest numbers that can be entered on the keypad.