CMPT 130: Week 4 Lab Work

1. Write a C++ program that reads ten integers using a **while** loop and then prints the sum of the numbers.

2. Write a C++ program that reads ten integers using a **while** loop and then prints the average of the numbers.

3. Write a program that reads in ten integer numbers using a **while** loop and that outputs the sum of all positive numbers, the average of all positive numbers, the sum of all negative numbers, the average of all negative numbers, the sum of all the numbers, and the average of all the numbers entered.

4. Modify your program in question #3 above so that to first ask the user how many integers to read (call this variable **n**) and then use a **while** loop to read **n** numbers and perform the computations.

5. What happens if the user enters **n <= 0** in question #4 above? Explain your answer in English language.

6. You are practising solving problems and your problem solving skills are getting better. Assume you are able to solve two questions on your first day, four questions on your second day, six questions on your third day…etc. That is at a given day, you are able to solve two more questions than the previous day.

   Write a C++ program that reads the number of days you have practiced and prints the number of questions you have solved in each day and the total number of questions you have solved at the end your practice season. Do not use a mathematical formula to compute the total number of questions solved. Instead you must use loop to compute the number of questions solved at each day and finally print the sum of all problems solved.

7. Write a program that reads in an integer **n**, asserts **n** is positive (that is the program uses a **while** loop structure to repeatedly ask the user for **n** until the user enters a positive **n**), and then reads **n** integers from the user using a **while** loop. Your program should then output the sum of all even numbers, the average of all even numbers, the sum of all odd numbers, the average of all odd numbers, the sum of all the numbers, and the average of all numbers entered. Every negative entry should be skipped and not used in any of the calculations.

8. Write a C++ program that asks user to enter a positive integer n and then prints the product of 1*2*3*4*...*n. Assume the user input n is positive.

9. Write a C++ program that reads a positive integer user input **n** and then that reads **n** user input integers and finally prints the maximum of all the **n** user input integers. Assume the user input value for **n** is positive. For example if the value of **n** is 5 and the numbers are -9, -3, -7, -2, -6 then your program must print the maximum value is -2.

10. Write a C++ program that reads a positive integer user input **n** and then that reads **n** user input integers and finally prints the minimum of all the **n** user input integers. Assume the user input value for **n** is positive. For example if the value of **n** is 5 and the numbers are 9, -3, -7, -2, -6 then your program must print the minimum value is -7.

11. Write a C++ program that reads a positive integer user input **n** and then that reads **n** user input integers and finally prints the maximum in absolute value of all the **n** user input integers. Assume the user input value for **n** is positive. For example if n is 4 and the user input are 2, -3, 4, -6 then your program must print **The maximum in absolute value is -6.** For example if the value of **n** is 5 and the numbers are 9, -3, -7, -25, -6 then your program must print the maximum in absolute value is -25.

12. Write a program that reads a positive integer number **n** and then prints the double data type value **y** given by:

$$y = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \cdots \pm \frac{1}{n}$$

Note that the last term may have a plus sign or a minus sign depending on the value of **n**. Use a **while** loop. Assume that the user input value for **n** is positive integer. Also note that the divisions must be performed in a float/double domain to avoid truncation.

13. Write a C++ program that reads a positive integer **n** and prints "The number is prime" if **n** is a prime number and prints "The number is not prime" if **n** is not a prime number. Assume the user input is an integer greater than 2. Hint:- use a **while** loop to count how many divisors **n** has between **2** and **n-1**. If it has zero divisors, it is a prime number; otherwise it is not a prime number.

14. Write a C++ program that reads a positive integer **n** and prints "The number is composite" if n is a composite number and prints "The number is not composite" if n is not a composite number. **Hint:-** Note that a number is composite if it is not prime; and vice versa. Use your ideas in Question #13 above.

15. The Fibonacci numbers are given as 1, 1, 2, 3, 5, 8, 13, 21, ... That is the first number is 1, the second number is 1 and after that each number is the sum of the previous two numbers. Write a C++ program that prints the first 20 Fibonacci numbers.

16. Write a C++ program that asks the user to enter a positive integer **n** and then that reads **n** user input integers and finally prints the message **The numbers you entered are in increasing order** if they are in increasing order or the message **The numbers you entered are NOT in increasing order** if they are not. See the remark below.
Remark:- Remember given the numbers $\mathbf{a_1, a_2, a_3, ..., a_n}$; we say they are in increasing order if $\mathbf{a_1 \leq a_2 \leq a_3 \leq .... \leq a_n}$. Moreover remember that if you have only one number from the user input then the number automatically is in increasing order.

17. Write a C++ program that asks the user to enter a positive integer **n** and then that reads **n** user input integers and finally prints the message **The numbers you entered are in strictly increasing order** if they are in increasing order or the message **The numbers you entered are NOT in strictly increasing order** if they are not.

Remark:- Remember given the numbers $\mathbf{a_1, a_2, a_3, ..., a_n}$; we say they are in increasing order if $\mathbf{a_1 < a_2 < a_3 < .... < a_n}$. Assume you have more than one number from the user input. Only one user input number does not make sense in this question.

18. Write a C++ program that asks the user to enter a positive integer **n** and then that reads **n** user input integers and finally prints the message **The numbers you entered are in decreasing order** if they are in increasing order or the message **The numbers you entered are NOT in increasing order** if they are not.

Remark:- Remember given the numbers $\mathbf{a_1, a_2, a_3, ..., a_n}$; we say they are in increasing order if $\mathbf{a_1 \geq a_2 \geq a_3 \geq .... \geq a_n}$. Moreover remember that if you have only one number from the user input then the number is automatically in decreasing order.

19. Write a C++ program that asks the user to enter a positive integer **n** and then that reads **n** user input integers and finally prints the message **The numbers you entered are in strictly decreasing order** or the message **The numbers you entered are NOT in strictly decreasing order** if they are not.

Remark:- Remember given the numbers $\mathbf{a_1, a_2, a_3, ..., a_n}$; we say they are in strictly decreasing order if $\mathbf{a_1 > a_2 > a_3 > .... > a_n}$. Assume you have more than one number from the user input. Only one user input number does not make sense in this question.

20. Write a C++ program that prints the following pattern using a **while** loop. You program must have only ONE **cout << " * ";** statement. Therefore you must use a loop.

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

21. Write a C++ program that that reads a positive integer value **n** from the user and then prints the following pattern using a **while** loop.

```
* * * * *
* * * * *
* * * * *
⋮
* * * * *
```

There should be **n** lines of output.

22. Write a program that scores a blackjack hand. In blackjack, a player receives three to five cards. The cards 2 through 10 are scored as 2 through 10 points each. The face cards - jack, queen, and king - are scored as 10 points. The goal is to come as close to a score of 21 as possible without going over 21. Hence, any score over 21 is called "busted". The ace card is scored as either 1 or 11 points, whichever is better for the user. For example an ace, a four and a six can be scored as either 11 or 21. Since 21 is a better score, this hand is scored as 21. An ace, a three, a five and queen can be scored as either 19 or 29. Since 29 is a "busted" score, this hand is scored as 19.

Write a complete C++ program that first asks the user how many cards the payer wants. Call this number **n**. In your program **n** must be between 3 and 5 both inclusive. If the user enters a number less than 3 or greater than 5, then your program must ask the user again and again and again for a value of **n** until a value between 3 and 5 both inclusive is entered.

Next, use a **while** loop to read **n** card values. Card values are 2 through 10, jack, queen, king, and ace. A good way to handle these inputs is to use the *char data type* so that the card input 2, for example, is read as the character '2', rather than as the integer 2. Input the values 2, 3, 4, ..., 9 as the characters '2', '3', '4', ... '9'. Input the values 10, jack, queen, king, and ace as the characters 'T', 'J', 'Q', 'K', and 'A'. For simplicity assume the letters entered will be only upper case letters.

After reading in the card values, your program should convert them from character values to numeric card scores, taking special care for aces. The output is either a number between 2 and 21 (inclusive) or the word Busted.

23. Repeat Q21 but this time instead of reading **n** card values from the user; assign each of the **n** cards a random card value among all the possible card values.