**1. Number average**

Write a program that lets the user enter exactly five integers. After all the numbers have been entered, the program should display the largest and smallest numbers entered and finally the average of these values (a double). Here is an example of execution of this program (user input is in bold):

You will enter a serie of five integers and I will then tell  
you the largest and the smallest numbers you entered  
  
Enter integer #1: **2**  
Enter integer #2: **8**  
Enter integer #3: **7**  
Enter integer #4: **-5**  
Enter integer #5: **4**  
  
The greatest number you entered is 8  
The least number you entered is -5  
The average of the five numbers you entered is 3.2

The average should be displayed with one decimal position.

**2. Exchange currency**

Write a program that lets the user enter a currency (a string) and an amount greater than zero (a double) and then display the equivalent amount in USD. Here is an exemple of execution of this program (user input is in bold):

Enter the currency (’euro’, ’vnd’, ’gpb’ or ’cny’): euro  
Enter the amount (in euro): 375  
  
375.00 euro = 406.19 usd

If the currency input by the user is wrong the program displays the appropriate error message:

Enter the currency (’euro’, ’vnd’, ’gpb’ or ’cny’): sgd  
  
Sorry, unknown currency

If the amount is less or equal to zero, the program displays the appropriate error message:

Enter the currency (’euro’, ’vnd’, ’gpb’ or ’cny’): vnd  
Enter the amount (in vnd): -100  
  
Sorry, the amount must be greater than zero

You can find current rates for the currencies shown on the example on the internet. These values should be defined as constants in your program.

**3. Body Mass Index**

Redo exercise 4 of lab #2 with the following changes: now the program is displaying the BMI **and** its interpretation, using the following table:

|  |  |
| --- | --- |
| **BMI range** | **Interpretation** |
| less than 15 | Very severely underweight |
| from 15.0 to 16.0 | Severely underweight |
| from 16.0 to 18.5 | Underweight |
| from 18.5 to 25 | Normal (healthy weight) |
| from 25 to 30 | Overweight |
| from 30 to 35 | Obese Class I (Moderately obese) |
| from 35 to 40 | Obese Class II (Severely obese) |
| over 40 | Obese Class III (Very severely obese) |

Here is an example of execution of this program (user input is in bold):

Enter the mass in kilograms: **75**  
Enter the height in meters: **1.78**  
  
The BMI is 23.7  
Interpretation: Normal (healthy weight)

Additionally, your program should be protected against bad input: if the mass or the height given by the user is less or equal to zero, your program should display the appropriate error message.

**4. Math repetitor**

Redo exercise 6 of lab #2 with the following changes: now the user can submit the answer of the proposed addition and the program will tell her/him if this answer is correct. Here is a log of execution of this program (user input is in bold):

Welcome to the Math tutor

  178  
+ 360  
----- Enter the answer below:  
  **538**  
  
Congratulations!  
The answer is 538

Here is another log of execution of this program:

  388  
+ 481  
----- Enter the answer below:  
  **769**  
  
Sorry that is incorrect.  
The answer is 869

You must start from your version of exercise 6 from lab #2 and try to adapt it to the new context.

**5. Bank charges**

A bank charges $10 per month plus the following check fees for a commercial checking account:

* $.10 each for fewer than 20 checks
* $.08 each for 20–39 checks
* $.06 each for 40–59 checks
* $.04 each for 60 or more checks

The bank also charges an extra $15 if the balance of the account falls below $400 (before any check fees are applied).  
  
Write a program that asks for the beginning balance and the number of checks written. Compute and display the bank’s service fees for the month. Do not accept a negative value for the number of checks written. If a negative value is given for the beginning balance, display an urgent message indicating the account is overdrawn.

**6. Date**

Write a program that reads a date from the user in numerical format and prints that date using names for the months.  Those are two logs of execution of this program (user input is in bold):

enter a date in the format dd/mm/yy: **25/12/2018**  
The date is: December 25, 2018

enter a date in the format dd/mm/yy: **1/4/2019**  
The date is: April 1, 2019

In case of bad input, the program stops with the suitable error message. Bad inputs are:

* a day less than one or greater than 31
* a month less than one and greater than 12
* a missing character '/' at the right place

In this exercise we don't take in account bad day depending on the number of days in the month. For example, 31/2/2018 is considered a valid date. Those are two logs of execution of this program (user input is in bold):

enter a date in the format dd/mm/yy: **13/15/2016**  
wrong month

enter a date in the format dd/mm/yy: **12/5:2018**  
missing '/' after the month

You can use the switch statement to solve this problem.