Aug 9, 2020

Programming Exercises

Level 1

Let the user enter a number. You output that many stars the user defined by a number.

Output:

```
Please enter a number: 5

*****
```

Level 2

Let the user enter a name. Insert that name inside a sentence and display it.

Output:

```
Please enter a name: Peter

On a stony ground Peter awaked from a dream.
```

Level 3

You will write a short visual novel. Ask the user two questions, it can be anything you want. Based on the user's answer the program branches. In the end you have four different endings for your short story. At the end, tell the user which ending he reached.

Output:

```
You walk around and come across an abandoned house. What do you do?
1: enter the house
2: turn around and go back where you came from
Select one of the above options: 2
A gloomy old lady blocks the way where you came from. What do you do?
1: Walk past the old lady
2: React shocked and just stare at her
Select one of the above options: 1
You walk past the old lady. You hear that she starts walking, you turn
around. The old lady walks inside her house.
```

```
Congratulations, you reached ending 3.
```

Level 4

Output all even numbers from 0 to 1000.

Output:

```
0 2 4 6 ... 996 998 1000
```

When you have the expected output, extend your program by counting how many numbers you outputted.

Output:

```
0 2 4 6 ... 996 998 1000
count: ? // What count do you expect, 500?
```

Level 5

You are a mage. You have 100 magic points (MP). You can cast one skill, "The Fireball". It consumes 15 MP. If you have insufficient MP, you are not able to cast your skill anymore. Please note that you shall also output the quotation marks -> " <-.

Output:

```
I am a mage, I have 100 MP and if you want me to cast "The Fireball"
enter yes: yes
I am a mage, I have 85 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 70 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 55 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 40 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 25 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 10 MP and if you want me to cast "The Fireball" enter
yes: yes
I do not have enough MP to cast "The Fireball".
```

Level 6

Let the user enter a number from 0 to 999. You output the same number, but you always want to display that number with three digits.

Output:

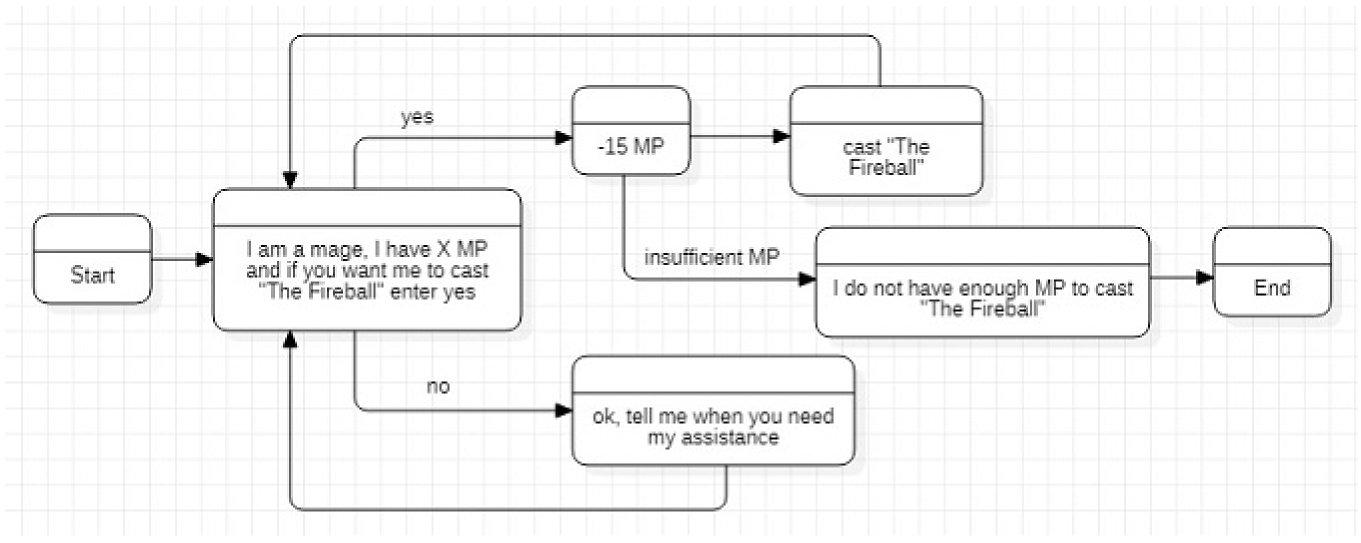
```
Please enter a number from 0 to 999: 3
003

Please enter a number from 0 to 999: 17
017

Please enter a number from 0 to 999: 127
127
```

Level 7

Return of "The Fireball". Did you notice that you only have programmed the yes case in Level 5? But what happens if you press no? Now you extend "The Fireball" program to also handle the input "no".


Start
I am a mage, I have X MP and if you want me to cast "The Fireball" enter yes
yes
-15 MP
cast "The Fireball"
insufficient MP
I do not have enough MP to cast "The Fireball"
End
no
ok, tell me when you need my assistance

In the upper part of the image you can see what happens if you enter "yes". That should be familiar. The lower part is new. It describes the "no" case. In That case you just loop back to "I am a mage, I have X MP ..." and do not subtract the MP.

Output:

```
I am a mage, I have 100 MP and if you want me to cast "The Fireball"
```

```
enter yes: yes
I am a mage, I have 85 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 70 MP and if you want me to cast "The Fireball" enter
yes: no
ok, tell me when you need my assistance.
I am a mage, I have 70 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 55 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 40 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 25 MP and if you want me to cast "The Fireball" enter
yes: yes
I am a mage, I have 10 MP and if you want me to cast "The Fireball" enter
yes: yes
I do not have enough MP to cast "The Fireball".
```

Level 8

Find the item which generates the most value over time. You are a weaponsmith. You can craft three items: Speer of Fate, Sword of Agony and Staff of Blessings. Each weapon takes time to craft, each weapon has a value for which you will sell it. We assume that if you crafted a weapon, you can sell it immediately. The question is: which weapon should you craft to gain the most money from it over time?

	Speer of Fate	Sword of Agony	Staff of Blessings
Needed time to craft	27m 03s	19m 46s	54m 31s
Selling price	23062	18728	41502

For further understanding: Imagine that there are three weaponsmiths. Weaponsmith A only crafts Speer of Fate, weaponsmith B only crafts Sword of Agony and weaponsmith C only crafts Staff of Blessings. Which one of them will be the richest after e.g. a month or which one will gain money faster. Or imagine: for a whole month you craft only one of the above weapons. With which weapon do you have the highest hourly earnings (german: Stundenlohn). E.g. with Weapon A you get 9.50€ per hour, with Weapon B you get 10.20€ per hour. Under the aspect that every weapon will sell immediately, you will only produce Weapon B, right?

For each weapon you shall calculate the *revenue score* which is comparable to hourly earnings. The revenue score indicates which one of the weapons is the most profitable to craft. In the end you sort your list of weapons by the revenue score and display it accordingly. #1, #2 and #3.

Output:

```
Speer of Fate
-------------
needed time to craft: 27m 03s
selling price: 23062 gold

Sword of Agony
--------------
needed time to craft: 19m 46s
selling price: 18728 gold

Staff of Blessings
------------------
needed time to craft: 54m 31s
selling price: 41502 gold

Revenue Scores // for example. Don't know if the order is correct. That is your job + the
-------------- // scores :)
#1 Speer of Fate
    revenue score: ??
#2 Sword of Agony
    revenue score: ??
#3 Staff of Blessings
    revenue score: ??
```

Level 9

Random damage. Did you notice in RPGs that you do not inflict damage with the exact value every time? You have min and max value wherein that damage value moves. So, we do the same thing now. Try to inflict damage between 1000 and 1500 including these values.

Output:

```
Enter 'attack' to deal some damage: attack
inflicted 1368 damage
Enter 'attack' to deal some damage: attack
inflicted 1233 damage
```

Level 10

Let the user enter a word and separate each letter with a space.

Output:

```
Please enter a word: hello
h e l l o
```

Level 11

Calculate 15 divided by 7.

Output:

```
15 / 7 = 2.14
```

Level 12

Create an enum named “Color” which contains “red”, “green” and “blue”. Output these enums. What does it output? Red, green and blue? Or something else? Try explaining the output and in which cases you would use an enum in general.

Output:

```
red = ?
blue = ?
green = ?
```

Level 13

Display 100 random numbers from 0 to 10. Output how often the 5 has been displayed. Also check if the 0 and 10 are contained under these numbers.

Output:

```
100 random numbers from 0 to 10:
3 9 3 1 0 9 3 4 0 8 0 1 4 9 3 10 0 1 3 4 10 4 2 3 3
6 2 10 9 7 9 0 6 10 1 6 8 2 8 7 10 6 6 1 4 9 0 4 9 1
6 6 6 6 9 6 0 9 6 8 3 1 8 9 1 7 3 7 9 9 1 8 6 8 9 7
4 7 10 0 6 6 6 0 1 2 6 10 0 9 7 4 9 4 0 10 0 3 4 7

5 is displayed 7 times.
```

Level 14

Testing your random numbers and check if the bounds are included. Let's bother you once more with random numbers. In games it is a lot more fun to get e.g. random items from a chest. That chest contains four items with the IDs 1, 2, 3 and 4. Would be pretty annoying if your random function would only pick 1, 2 or 3 because you set your bounds incorrectly. So in that exercise we will do two things 1) program an item lottery and 2) test the reliability of it.

1) The lottery. You can win one of four items. Item 1: A shuriken with a chance of 25%. Item 2: A banana with a chance of 49%. Item 3: Magic Gloves with a chance of 25%. Item 4: A Coupon for free drinks for 24h with a chance of 1%.

2) testing your lottery. Can you get all four items? This 1% item is pretty hard to get. Is it just bad luck or is it impossible to get, because the lottery has a bug? To test the lottery, we call the lottery function many times and check if item 4 gets chosen sometimes. 1% means we pick that item 1 time out of 100 times. So we call that function 1000 times or 10000 times whatever you feel more comfortable with. Item 4 must be chosen at least one time after that many times to know that you set the bounds correctly. In a second step we check if the percentage value is about correct. So, out of 1000 times, Item 4 needs to be picked round about 10 times, with 10000 tries it should be picked around about 100 times. Write such a test function and see if the percentage values are about right. Therefore you display after the test run how often which item has been picked and what percentage value that corresponds to.

Output:

```
The lottery has four items:
---------------------------
1: Shuriken, chance 25%
2: Banana, chance 49%
3: Magic Gloves, chance 25%
4: Coupon for 24h free drinks, chance 1%

What do you want to do (1: try my luck on lottery, 2: test lottery): 1
You got: Banana. Congratulations!

What do you want to do (1: try my luck on lottery, 2: test lottery): 2

Running the lottery for 1000-times
-----------------------------------
Shuriken has been picked 243 times, that corresponds to 24.3%
Banana has been picked 495 times, that corresponds to 49.5%
Magic Gloves has been picked 257 times, that corresponds to 25.7%
```

```
Coupon has been picked 89 times, that corresponds to 8.9%
```

Level 15

Encrypting and decrypting text. The user has to choose from two options: 1) encrypt text 2) decrypt text. Encrypting means e.g. taking the text “abcde” and convert it to “hijkl”. Did you notice that we shift each letter by 7 forwards inside the alphabet? When decrypting we do the same, only backwards. So the user enters “hijkl” and we convert it to “abcde”. That procedure is called Caeser Cipher (dt: Caeser Verschlüsselung). The 7, how much you shift each letter, is the encryption key. The user shall also be able to choose an encryption. So, you let him also enter that key, which is just a number, when encrypting and decrypting a text.

Output:

```
What do you want to do (1: encrypt text, 2: decrypt text): 1
Enter your text: abcde
Encryption key (a number): 7
Your encrypted text is: hijkl

What do you want to do (1: encrypt text, 2: decrypt text): 2
Enter your text: hijkl
Decryption key (a number): 7
Your decrypted text is:  abcde
```

Did you think about, if you actually type in a whole text with many words. How is that space character handled? How are german umlauts (ä,ö,ü,ß) handled? Does your encryption/decryption still work? Try explaining why it still works or why it does not.

Level 16

Huuh, ok, let's get more chilled now. Try creating a program that crashes with e.g. an “access violation” error, “stack overflow” error or an “windows had to close that program” error. What is an “access violation”? What is a “stack overflow”? Can you create a program, that produces a blue screen? Is that possible? Try explaining why it is possible or why it is not.

Level 17

Write a program with which you get the CPU usage to 100%.

Level 18

Write a program with which you get the RAM usage to over 90%. What happens when it reaches that mark? Why does it not go to 100%?

Level 19

Create a list of words with upper and lower case words. Sort that list ascending independent of their case (lower/upper case). So, the list starts with words which begin with A, and ends with words that end with Z.

Level 20

You create a program that detects the language of a text. The user can enter a word or a sentence and you tell the user from which language that word or sentence origins. Your program should also detect Chinese, Japanese, Korean, Thai and Arabic language. Therefore, your program must understand these characters. After processing the text, you tell the user how sure you are what language it is. E.g. the user entered 10 words, you detected 9 words as English and 1 as unknown, then you would tell the user: "I am 90% sure that your text is English language".

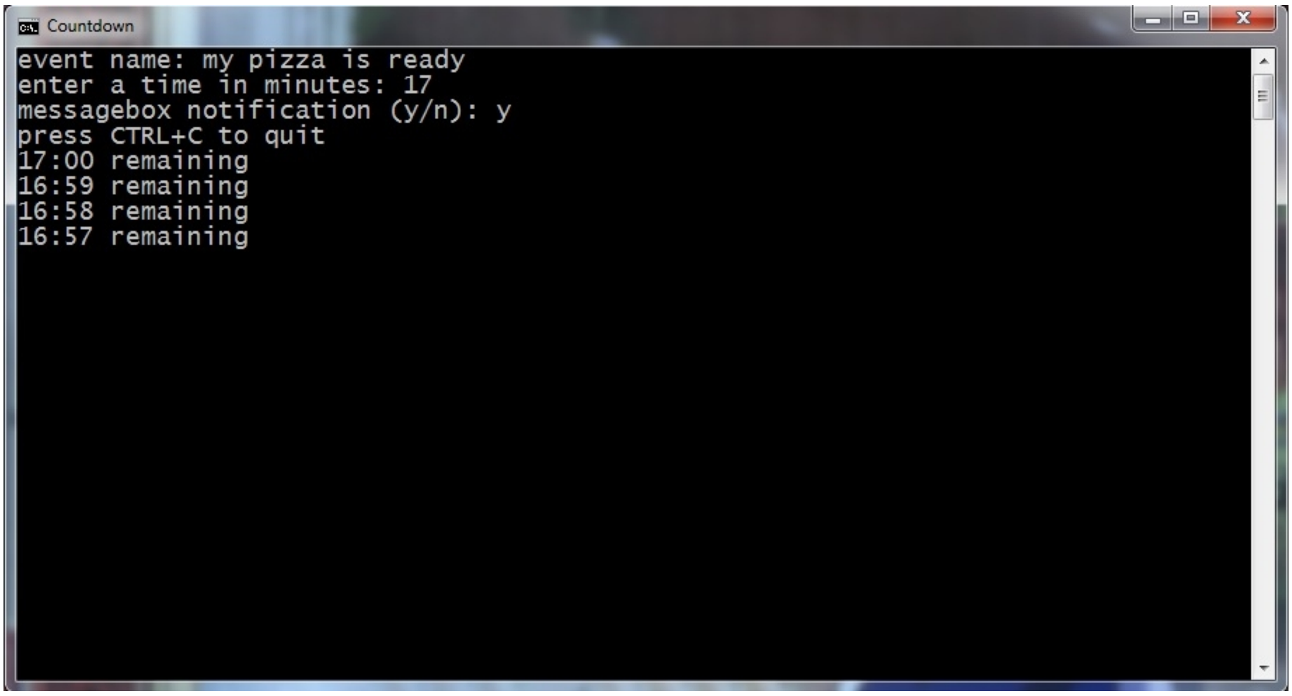
You will create a user interface (UI) where you can paste various texts into an input field, let's call that UI element *userInputField*. A text field above the input field will tell the user, what the program is currently doing, let's call that UI element *statusTextField*. The detection my take some seconds, so the user wants to know that the program is currently running and what exactly it is doing. So you e.g. tell the user which language you are currently checking what the text in the *userInputField* could be. Display a "processing animation" when your program does some work. An animation which you can often observe in programs when you have to wait for a response. It is mostly a *spinning circle*. After entering a text in the *userInputField*, your program starts running automatically. There is no button to click start.

Your program should also detect abbreviations such as lol, rofl, lmao, brb, ty, thx, np and all others which you can think of. For these abbreviations you should detect from which language these come from. Also think of abbreviations in your native language. Use Google to find as much as possible abbreviations and create an extra dictionary for them.

Detecting smileys. Your program should also detect smileys such as :), :D, :(, xD, ^.^, www, UvU, T^T, O.o and anything else you can think of. Use Google to find as much as possible smileys and create an extra dictionary for them.

Level 21

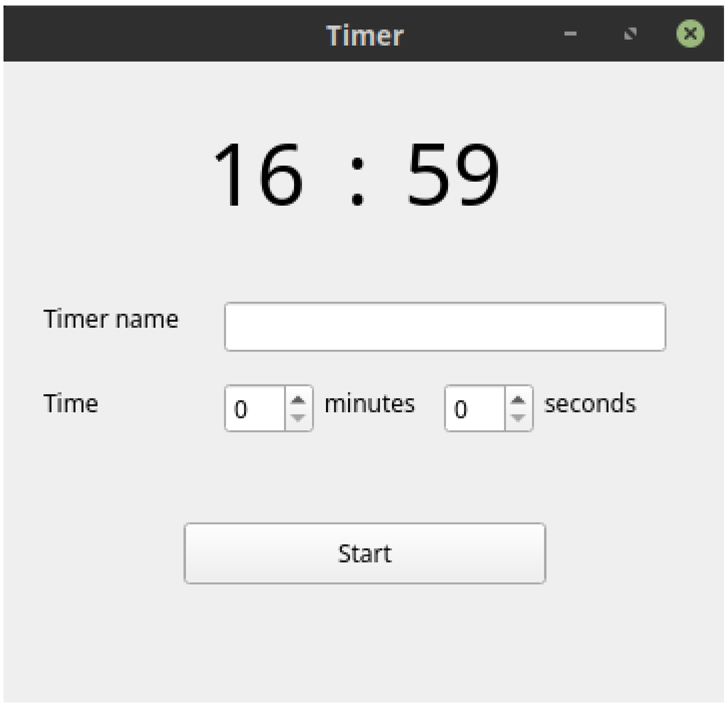
You create a timer application. The user enters a timer name, a duration in minutes and how the user should be notified when the timer reaches zero. For the notification the user can choose between console notification and pop-up notification.



```
event name: my pizza is ready
enter a time in minutes: 17
messagebox notification (y/n): y
press CTRL+C to quit
17:00 remaining
16:59 remaining
16:58 remaining
16:57 remaining
```

Level 22

You create again a timer application. This time it is not a console application. You will create a GUI application (graphical user interface). The user enters again a timer name and a time in minutes, additionally with seconds. The option to choose a notification type will be removed. It will always be a pop-up notification. The clock in the top shows the remaining time after starting the application.



Level 23

Registration and login. You create two console applications. A client application and a server application. You don't really create a server application, but it will behave like one. The client can register a new account in the server application. Therefore, the server application expects a user name and a password. After registration you can log into the server application. The server then grants you access to retrieve the current date and time.

More details on r*egistration*. The client hashes the password the user enters and only sends the hashed password to the server application. Like this, the server will never know the actual password. The user name will not be hashed. The server application saves the user name and the (hashed) password in a file accounts.txt.

More details on *login*. When logging into the server application, the user enters his user name and his password. The client application sends the user name as is, raw, and the password hashed to the server. The server checks if the user name exists and then checks if the password is correct. If anything is incorrect, you print “user name and/or password is incorrect”. For safety reasons you do not tell if that user exists you want to login or if the password is incorrect.

More details on *sending information to the server*. The client is a separate application and the server is a separate application. When the client wants to send information to the server, the client calls the server application with arguments. “server.exe <action> <user_name> <password>”. The server application starts and expects three arguments. Argument 1 “action”: you pass either “reg” for registration or “login” to login into an existing account. If it is neither of them, the server application prints “invalid action” to the console and quits. With the action “reg” you create a new user in accounts.txt from the following argument 2 “user_name” and argument 3 “password”. With the action “login” you check if argument 2 “user_name” is an existing user name in your accounts.txt. If it exists you check the password with argument 3 “password”. If it equals, you print “successful login” on the console. In any other case you print “user name and/or password incorrect” on the server application's console. Only the client sends data to the server. There is no backwards communication. The server never sends data. He only prints responses on his console. You can not simply start the server by double-clicking on its executable. He always needs three arguments to run. In any other case the application will quit immediately.

More details on the *client*. You can start the client application without arguments. That means you can just double click the executable and the client starts. When the client starts, it asks the user if he wants to register or to login. In both cases the client asks for a user name and a password. Before sending the user credentials, the client asks “Do you agree that I will send your user credentials to the server? (y/n)”. If you enter “y” the client will call the server application with the according information. If you enter “n” the client will abort. The client loops and infinitely asks “Do you want to register or to login. [1] register. [2] login?”.

Level 24

Item Shop console application where you play the adventurer. You earn money from adventures. You buy item such as Weapons, Armors, Potions etc. from an Item Shop. If you want to heal, because you got injured from an adventure, you want to buy a potion in the Item Shop.

On *an adventure* you have three normal monster fights. If you defeated all of them you can decide if you want to do the boss battle. You do not have to. You can go back and grind the three monsters again, go to the Item Shop, buy better equipment and when you feel ready, you can challenge the boss. If you loose any fight it is game over and you have to start all over again.

The adventurer's strength is on one hand defined by the *equipment* he is using. He can equip a sword, chest armor, leg armor, shoes, gloves and a hat. Better sword or gloves lead to increased attack. Better chest armor, leg armor or hat lead to increased defense. Better shoes lead to increased agility. Higher attack leads to more damage against monsters. Higher defense leads to less damage dealt from monsters. Higher agility leads to be able to attack multiple times. So you can be able to attack more than one time in a row, before the monster can attack

The adventurer's strength on the other side is defined by his *character level*. The character level only increases agility and evasion. Agility has been explained above and can also be increased by shoes. Evasion can only be increased by the character level. With more experience you become faster in what you are doing. That means you will be able to attack more often and you can evade attacks from monsters.

After each *fight* you get money from the defeated monsters. From boss monsters you get money, plus an item drop. The item drop is either a weapon or an armor.

In an *inventory* you can either equip items or use items. Equipping items means, you equip weapons and armors. Using items means, you can use items to e.g. heal yourself. In the inventory you can also see the character status. That means you get an overview what the adventurer is using as equipment, plus you see how your equipment influences your attack, defense and agility.

When you exit the game, the game state will be automatically saved in a *save file*. That means, all your items in the inventory and your progress on adventures will be saved. When you start the game, your save file will be loaded. That means you can continue where you stopped last time.

The *item shop* has more items available for selling the more you progress in adventures.

Level 25

HP bar. Health points bar like in an RPG.

Level 26

Generate a 3×3 magic square randomly. Write a function isMagicSquare which verifies the correctness of your generated magic square. What is a magic square? → https://en.wikipedia.org/wiki/Magic_square

Aug 9, 2020

Level 27

Write a command line application named *pwhash*. It takes a word as argument, hashes it with SHA-256 and outputs the hash. Search for a library, so you can use SHA-256 within your application.

Output:

```
$ pwhash password123
ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f
```