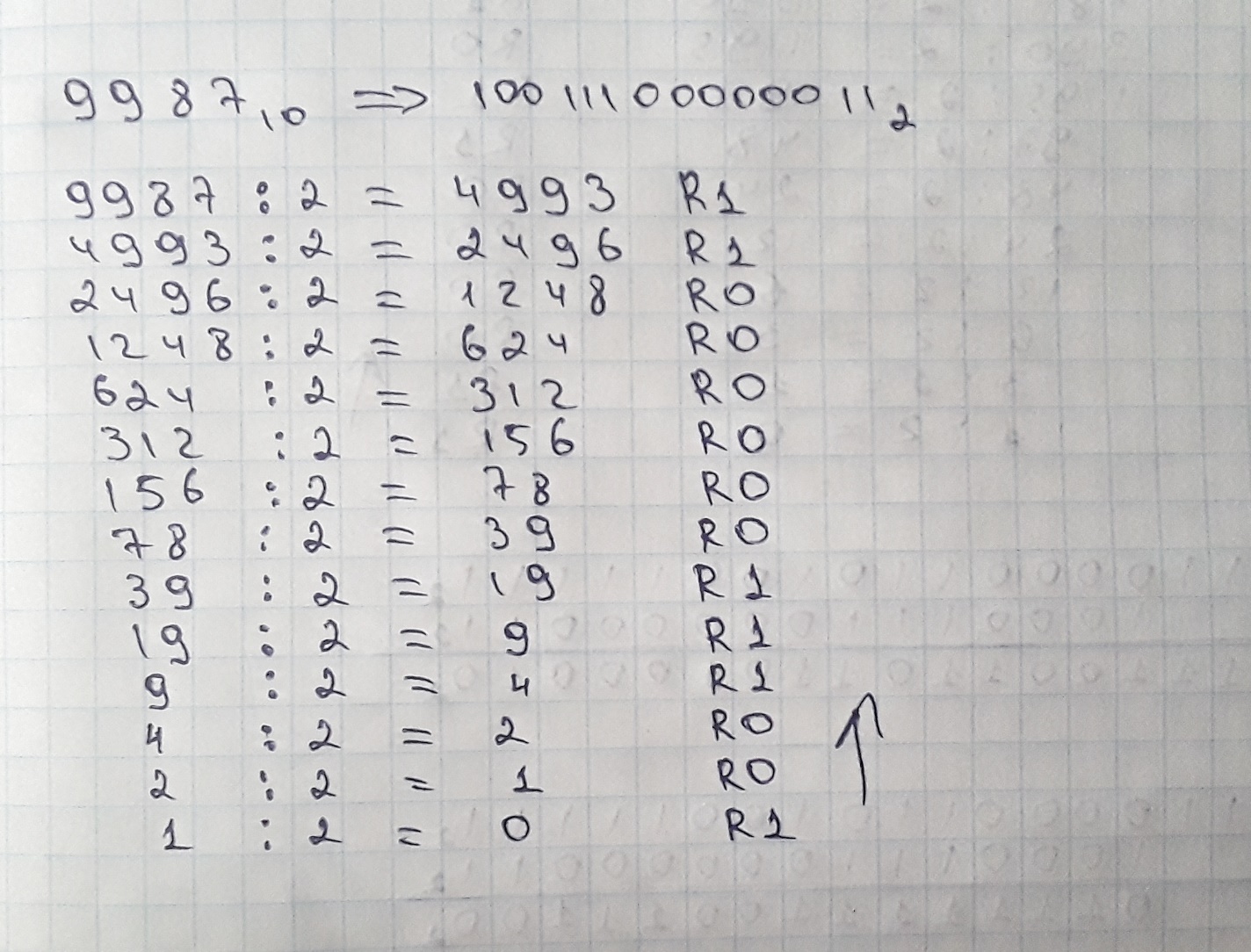
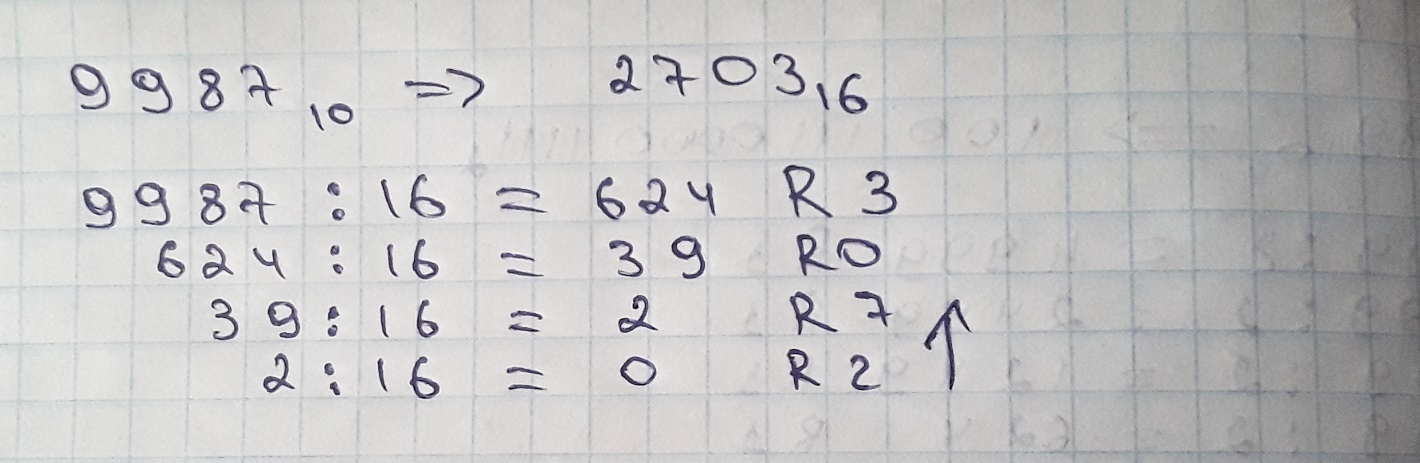
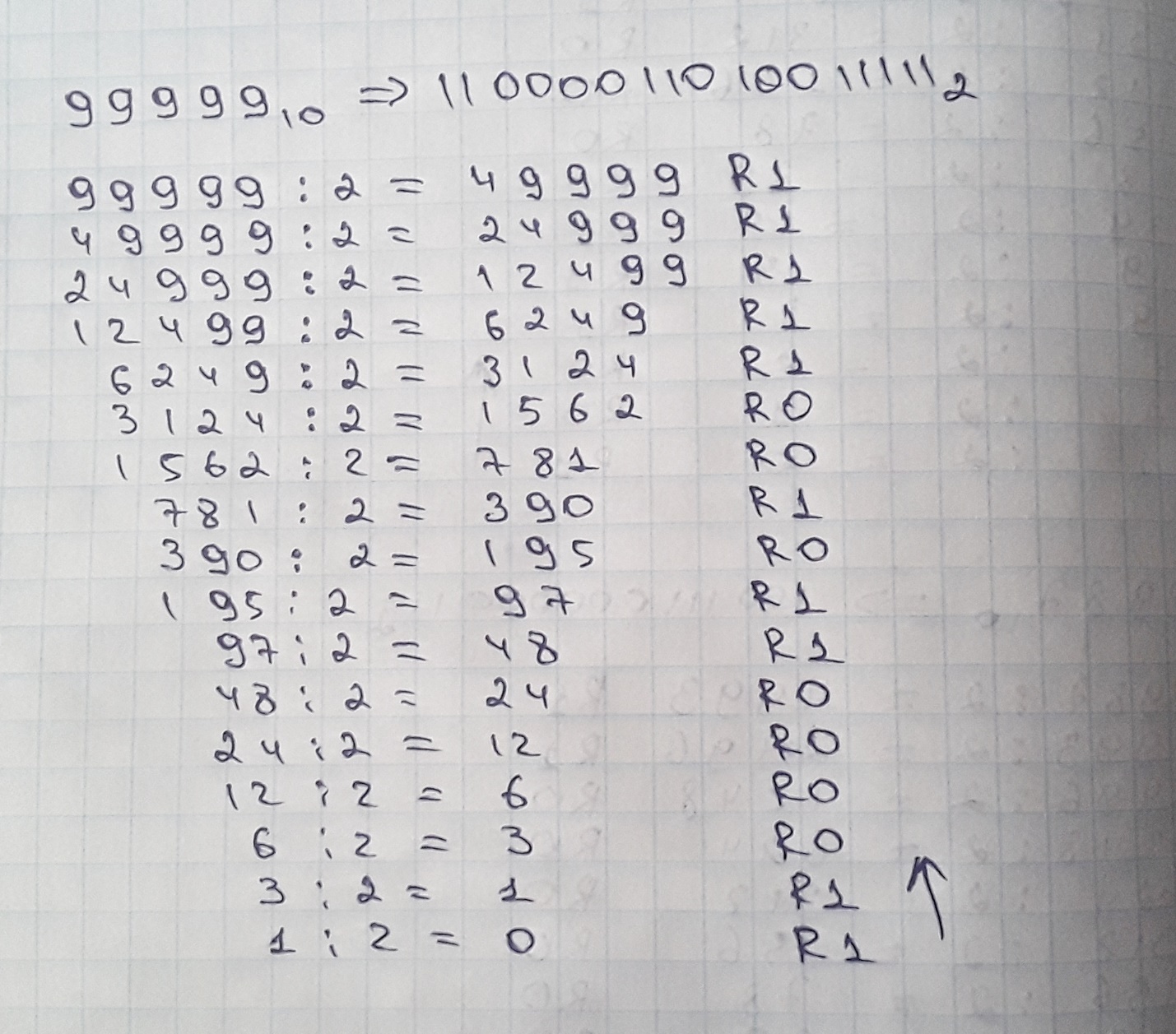
**Task 2**

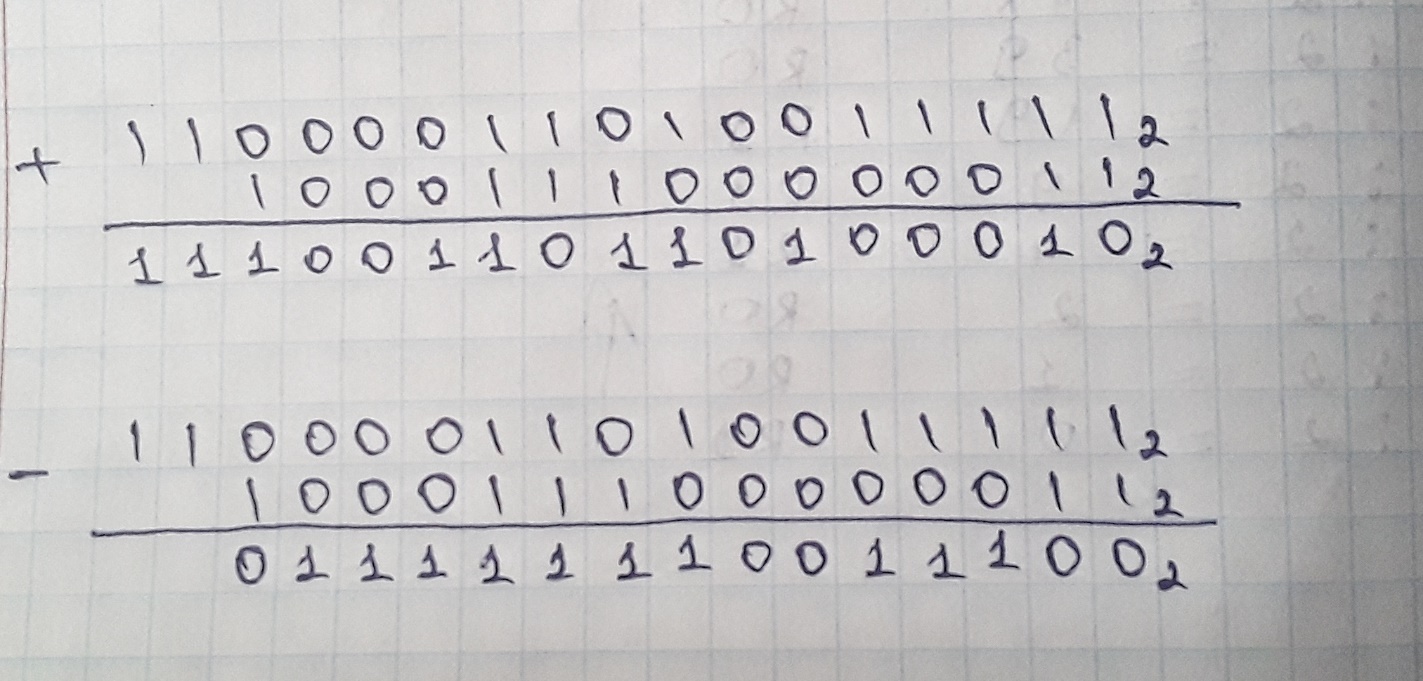
**a)** *Decimal to Binary (R – Remainder):*

*Decimal to Hex:*

**

**b)** *Conversion of 99,999 to binary:*

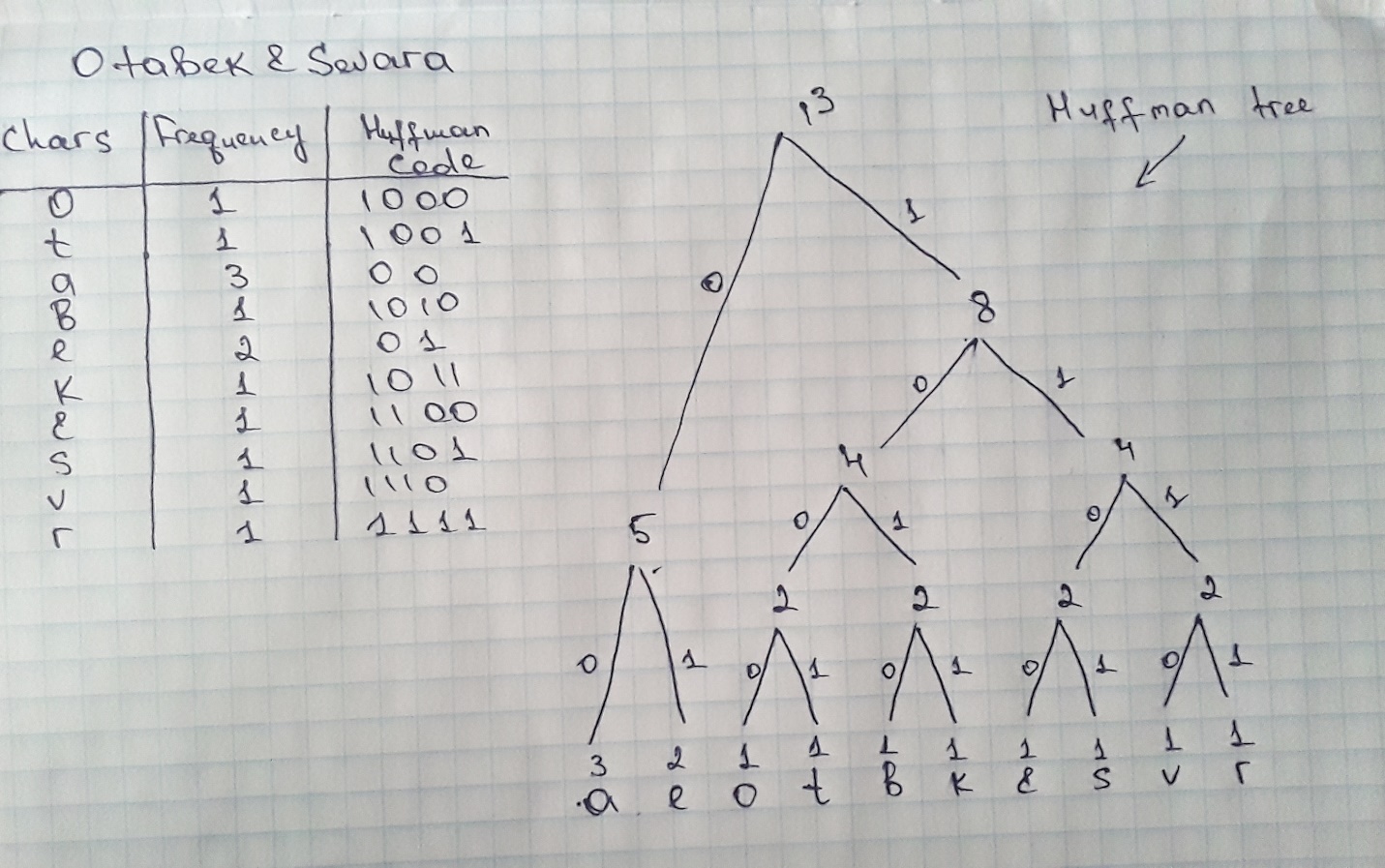
*Addition and subtraction, respectively: NEED MORE EXPLANATION HERE*

**

**c)** There are several reasons why using hexadecimal is more efficient than other number bases. The reasons are, firstly, since digits that are more closely look alike to usual base-10 number system used in hexadecimal, it is easier to read the digits at the first glance. Secondly, higher information density, that is in order to show any number between 0 and 255, only 2 digits are required in hexadecimal, and when it comes to do the exact same thing in binary, 8 digits are required (Savas, 2016).

**Task 3**

*Table and Huffman tree, respectively:*



**Before:** Otabek&Sevara = 12 characters \* 8 bits = 96bits

**After:** 100010010010100110111100110101111000111100 = 42bits

**Compression Ratio:** 42 / 96 = 0.4375

**Task 4**

**Step 1:**

The numbers are 9, 9, 8, 7, 4, 5, 2, 3, 7

In ascending order: 2, 3, 4, 5, 7, 7, 8, 9, 9

The number to look for is 4 (x = 4).

**Step 2:**

* Midpoint is 7 (9 / 2 = 4.5, 5th number is the midpoint).

2, 3, 4, 5, 7, 7, 8, 9, 9

* Since x < midpoint, ignore the range on the right (7 – 9).

2, 3, 4, 5, ~~7, 7, 8, 9, 9~~

* Midpoint is 3 (4 / 2 = 2, 2nd number is the midpoint).

2, 3, 4, 5, ~~7, 7, 8, 9, 9~~

* Since x > midpoint, ignore the range on the left (2 – 3).

~~2, 3,~~ 4, 5, ~~7, 7, 8, 9, 9~~

* Midpoint is 4 (2 / 2 = 1, 1st number is the midpoint).

~~2, 3,~~ 4, 5, ~~7, 7, 8, 9, 9~~

* x is equal to midpoint (5 = 5). The number we are looking for is found.

**Task 5**

**1.** In paged memory technique, main memory is split into parts or chunks of storage called frames. A frame is a physical memory unit with the fixed size, and frames are space of storage for pages. Pages are the sections or parts of a task or process (which is being executed), and depicts a logical memory unit.

The functionality of splitting a task or process into sections eases the difficulty of loading a task from locating an accessible huge section of space to locating sufficient small sections. This way a process or task does not have to be stored contiguously in memory anymore, and it is one of the advantages of paging.

A significant enlargement to the paged memory management is the concept of demand paging that favors from the case that not all sections of a program need to be stored in memory at simultaneously. Since the CPU can access only one page of a task or process at any time, it is not a big deal if the memory is allocated evenly to the other pages of that task.

The demand paging method support the concept of virtual memory, the idea that the size of a program is not restricted. With the other management techniques, the full program needs to be stored into memory as a continuous whole. What demand paging does is to remove that restriction (Dale and Lewis, 2002).

**2. a)** In order to produce a physical address, firstly, the frame number should be found with help of the page in the page-map table. Secondly, frame number should be multiplied by frame size and added to the offset to find the physical address.

The physical address of <2, 85> with the frame size of 1,024:

5\*1024 + 85 = 5,190.

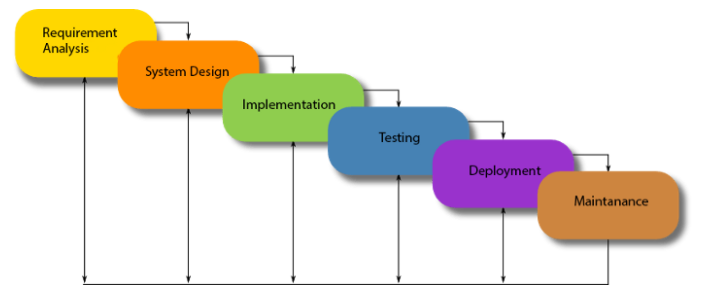
**b)** The physical address of <0, 1026> can not be found since the offset (1026) is greater than the frame size (which is 1024).

**Task 6**

Software development methodology is a process or series of processes used in software development. Two of the models, Waterfall and Agile, are going to be discussed below.

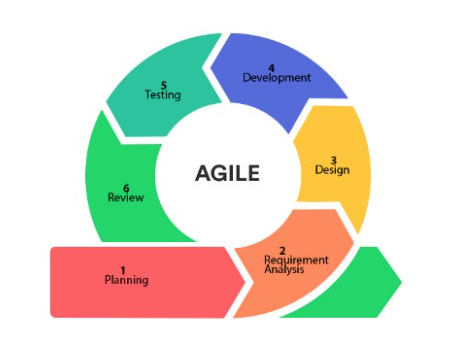
**Waterfall Model**

The model depicts the software development in a linear step-by-step flow. What this means is that every step in the development starts if the previous step is fully completed, and also overlapping in the steps are not possible.



**Agile Model**

The model consists of incremental and iterative process models in order to focus on adaptability of the task and customer satisfaction with the help of fast delivery of the software product. Agile model splits the process into small incremental sections, and these sections will be iterations. Every iteration period is usually ranging between 1-3 weeks, and it involves various teams working at the same time on different areas, namely planning, requirement analysis, design, development, testing, and review. When the iteration ends, the software product will be shown to the customers.



**Main differences between Waterfall and Agile Models:**

* Waterfall is a linear step-by-step flow while Agile is a repeated iteration of software development process
* Agile model is popular for its flexibility and adaptability whereas Waterfall is known for its structured methods.
* In Agile, testing occurs on the development phase while, in waterfall, testing happens after the implementation phase of the software.
* It is not possible to alter the requirements of the project in Waterfall once the development cycle starts whereas Agile grants changes in the process.

**When to use one of the models:**

**Waterfall:** Since it is one the effortless methods to manage, and each step has exact review process, this model works better for smaller projects where the instructions and requirements are easy to understand. Additionally, the process of the development and results are documented well, and with Waterfall shifting teams can be adapted easily.

**Agile:** One of the advantages of this method is that it is focused more on client process, and therefore it assures clients also are involved in the development process at the same time, and it is suitable for working with middle-sized, customer-based projects. Moreover, the software development process is fully built on incremental progress, and thus clients and team decide exactly what to do and what not, which result in reducing the risk in the process (Sandeepa, 2020).

**Task 7**

**Task 8**

**Void function example:**

*# Void functions are created to perform a particular task without returning*

*any values*

*# receiving user inputs and converting x and y to integer*

x = int(input('First number: '))

y = int(input('Second number: '))

operator = input('Operator (type +, -, \*, /):')

*# x, y, and operator below are arguments*

def simpleCalculator(x, y, operator):

*# according to user's provided operator, calculations will be performed*

*if* operator == '+':

*# showing result to the user*

        print(x + y)

*elif* operator == '-':

        print(x - y)

*elif* operator == '\*':

        print(x \* y)

*elif* operator == '/':

*# validating division*

*if* y == 0:

            print(f'{x} can not be divided by 0')

*else*:

            print(x // y)

*else*:

*# validating operator*

        print('Please enter only +, -, \*, or /')

*# invoking the function*

simpleCalculator(x, y, operator)

**Value returning function example:**

*# a return is a value that a function returns to the calling script or*

*function when it completes its task.*

*# receiving user inputs and converting x and y to integer*

z = int(input('First number: '))

t = int(input('Second number: '))

operator2 = input('Operator (type +, -, \*, /): ')

def calculator(z, t, operator2):

*# according to user's provided operator, calculations will be performed*

*if* operator2 == '+':

*# receiving sum of numbers*

        result = sumNums(z, t);

*elif* operator2 == '-':

*# receiving difference of numbers*

        result = subtractNums(z, t)

*elif* operator2 == '\*':

*# receiving multiplication of numbers*

        result = multiplyNums(z, t)

*elif* operator2 == '/':

*# receiving division of numbers*

        result = divideNums(z, t)

*else*:

*# validating operator*

        print('Please enter only +, -, \*, or /')

*# showing user’s the calculation result*

    print(result)

*# Helper Functions that return values*

def sumNums(z, t):

*return* (z + t)

def subtractNums(z, t):

*return* (z-t)

def multiplyNums(z, t):

*return* (z \* t)

def divideNums(z, t):

*# validating division*

*if* t == 0:

*return* (f'{z} can not be divided by 0')

*else*:

*return* (z // t)

*# invoking the function*

calculator(z, t, operator2)

**Reference list**

Dale, N. and Lewis, J. (2002). *Computer Science Illuminated,* edition. Sudbury, MA: Jones and Barlett Publishers.

Sandeepa, P. (2020). Waterfall Model vs Agile Model. *Medium*. Available from <https://medium.com/linkit-intecs/waterfall-model-vs-agile-model-4b96491bb9bf> [Accessed 14 January 2021].

Savas, N. (2016). Why do we use hexadecimal? *Medium*. Available from <https://medium.com/@savas/why-do-we-use-hexadecimal-d6d80b56f026> [Accessed 14 January 2021].