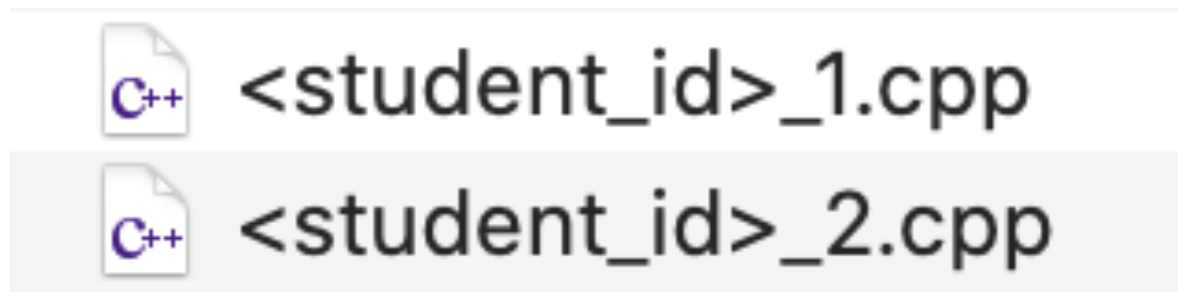
Week 5 Homework

Computer Programming Lab

2020/10/13

Remind

- 抄襲一律 0 分 （包含被抄襲者）
- 繳交期限：10/18(Sun.) 11:59 p.m.
- 繳交的檔案格式、名稱請符合以下規定
  - 請繳交 zip檔至 Ceiba作業區，名稱為 <student_id>.zip
  - 解壓縮後須符合下圖的格式、名稱



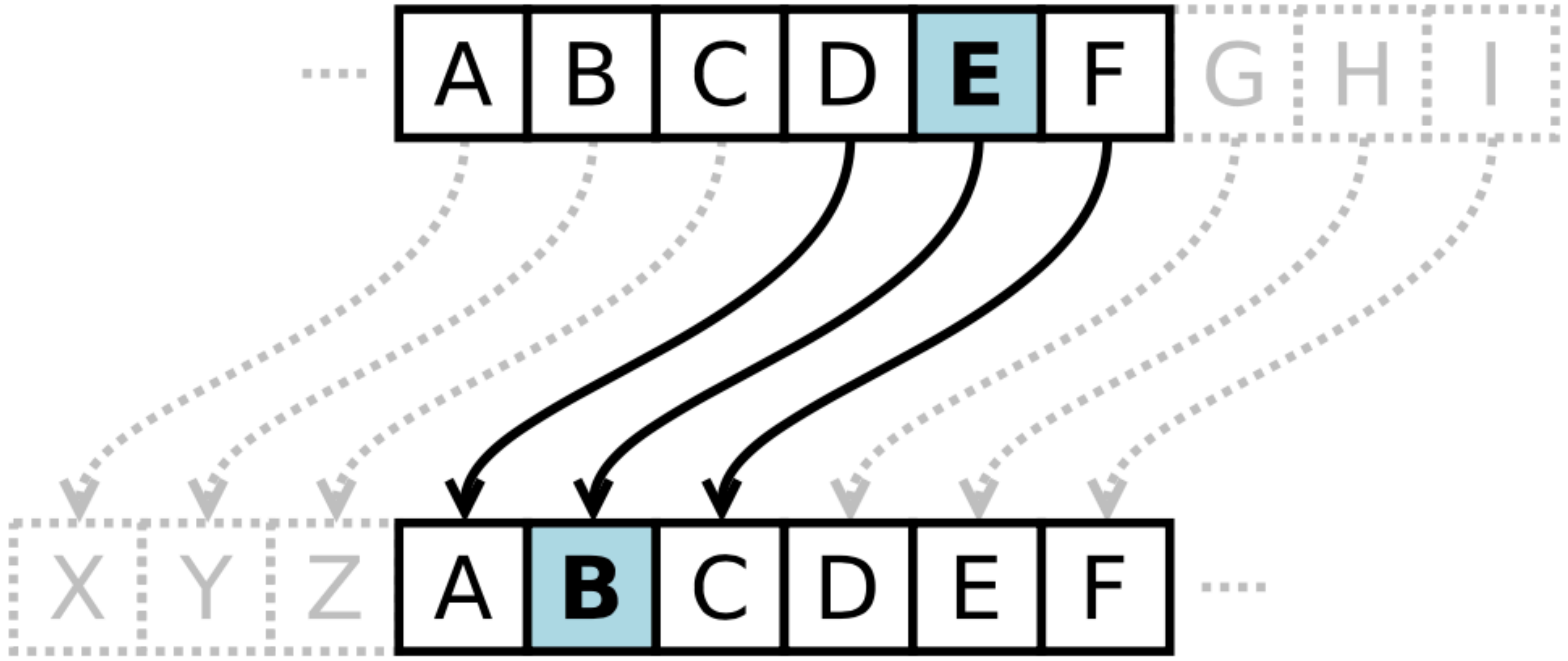
<student_id>_1.cpp
<student_id>_2.cpp

  - e.g. b12345678.zip
- 必須完成 Demo 才可以提早離開
- 若沒有完成 Demo 就中途早退，視同缺席

Problem 1 - Caesar Cipher (0.5%)

Description

Caesar cipher is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on.



Problem 1 - Caesar Cipher (0.5%)

Input

The first line contains an integer which indicates the shift. The negative number indicates the left shift, otherwise, the positive number indicates the right shift.

The second line is the string in lowercase. The length of the input string will be less than 100.
If the string contains a whitespace, output the whitespace directly.

Hint: Use `scanf` or `cin.get()` to deal with the whitespace

Output

The program should output the result of encryption.

Problem 1 - Caesar Cipher (0.5%)

Sample Input

```
3
happy coding
```

Plain Text

Sample Output

```
kdssb frglqj
```

Plain Text

File name

{Student_ID}_1.cpp

Problem 2 - Perfect Number (0.5%)

Description

In mathematics, perfect number is a term that denotes an integer equal to its sum of divisors excluding itself. For example, observe that $6 = 1 + 2 + 3$. As a result, 6 is a perfect number. On the countrary, $10 \neq 1 + 2 + 5$. So 10 is not a perfect number.

For this problem, you should implement a program that consumes an positive integer, and tells you whether it is a perfect number or not. If it is a perfect number, your program should output "perfect number". Otherwise, it should output "imperfect number".

Sample input

```
25
```

Plain Text

Sample output

```
imperfect number
```

Plain Text

File name

{Student_ID}_2.cpp

Problem 3 - Help Roger calculate his study time (1%)

Description

Roger is studying, and he wants to know how long he has been studying. He lists out many intervals of his study time, and he wants to calculate how much time he has used for studying. However, Roger's math is too poor to accomplish the task, so you should help him to do this. Given indefinite number of time intervals, and they are given in time sequence, calculate the total amount of time between all of the time intervals.

Problem 3 - Help Roger calculate his study time (1%)

Input

Time intervals are composed of starting time and ending time, and there is a "~" between starting time and ending time. Note that if ending time is smaller than starting time, it means that it has passed 0:00:00, and the ending time is at the next day. There will be an "EOF" at the end of the input.

Hint: You can use `scanf`

Output

The total amount of time between all of the time intervals. Note that your output should be this form: (hours:minutes:seconds), minutes and seconds should contain 2 digits. If minutes and seconds are smaller than 10, they should contain a leading zero. For example, if the total amount of time is 12 hours 36 minutes 6 second, the output should be 12:36:06.

Problem 3 - Help Roger calculate his study time (1%)

Sample input

```
5:00:00~8:00:00
12:00:00~14:00:00
19:00:00~22:00:00
23:00:00~3:00:00
```

Plain Text

Sample output

```
12:00:00
```

Plain Text

File name

{Student_ID}_3.cpp

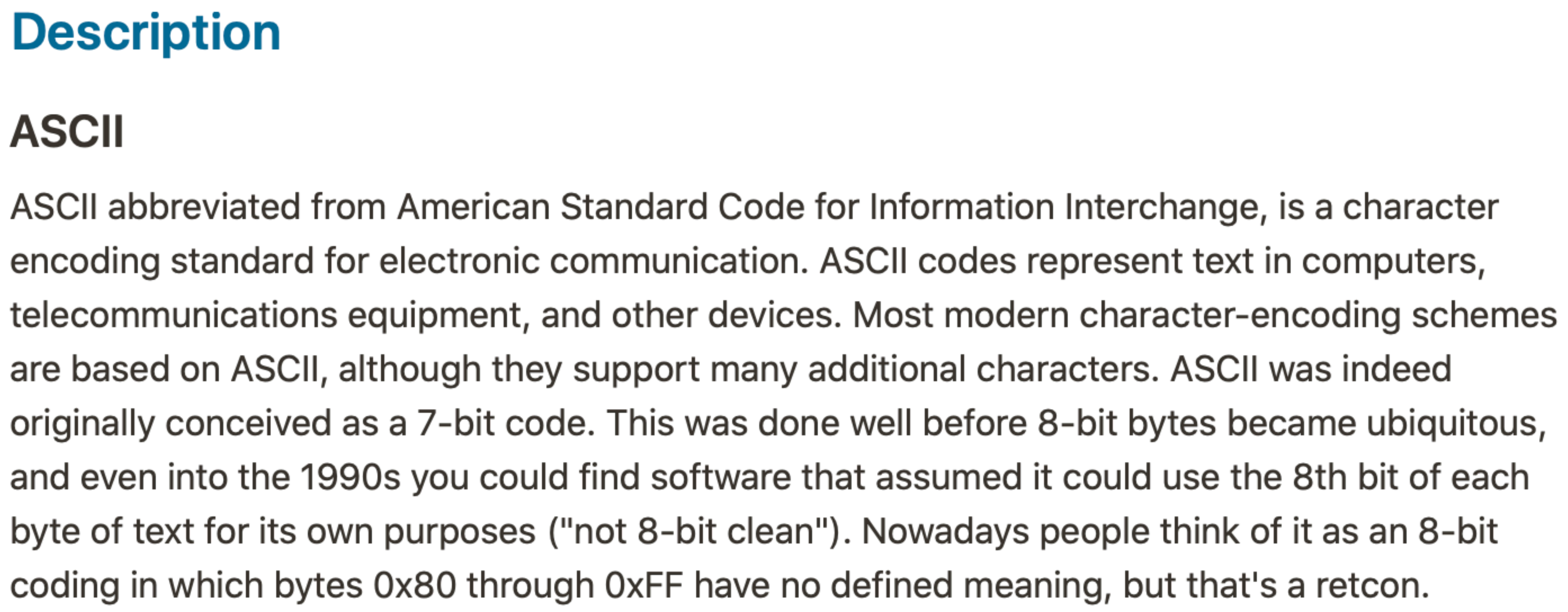
Problem 4 - Base Conversion (1.5%)

Description

Number Bases

A number base is the number of digits or combination of digits that a system of counting uses to represent numbers. A base can be any whole number greater than 0. The most commonly used number system is the decimal system, commonly known as base 10. Its popularity as a system of counting is most likely due to the fact that we have 10 fingers. Binary is the most commonly used non-base 10 system. It is used for coding in computers. Binary is also known as Base 2. This means it is composed of only 0's and 1's. For example 9 in binary/base 2 is 1001. Let's see how this works. Base 16, also known as the hexadecimal system, another common base when coding and using computer systems. In this case, we use the digits 0–9 and the letters representing two digits A(10), B(11), C(12), D(13), E(14), and F(15).

Problem 4 - Base Conversion (1.5%)

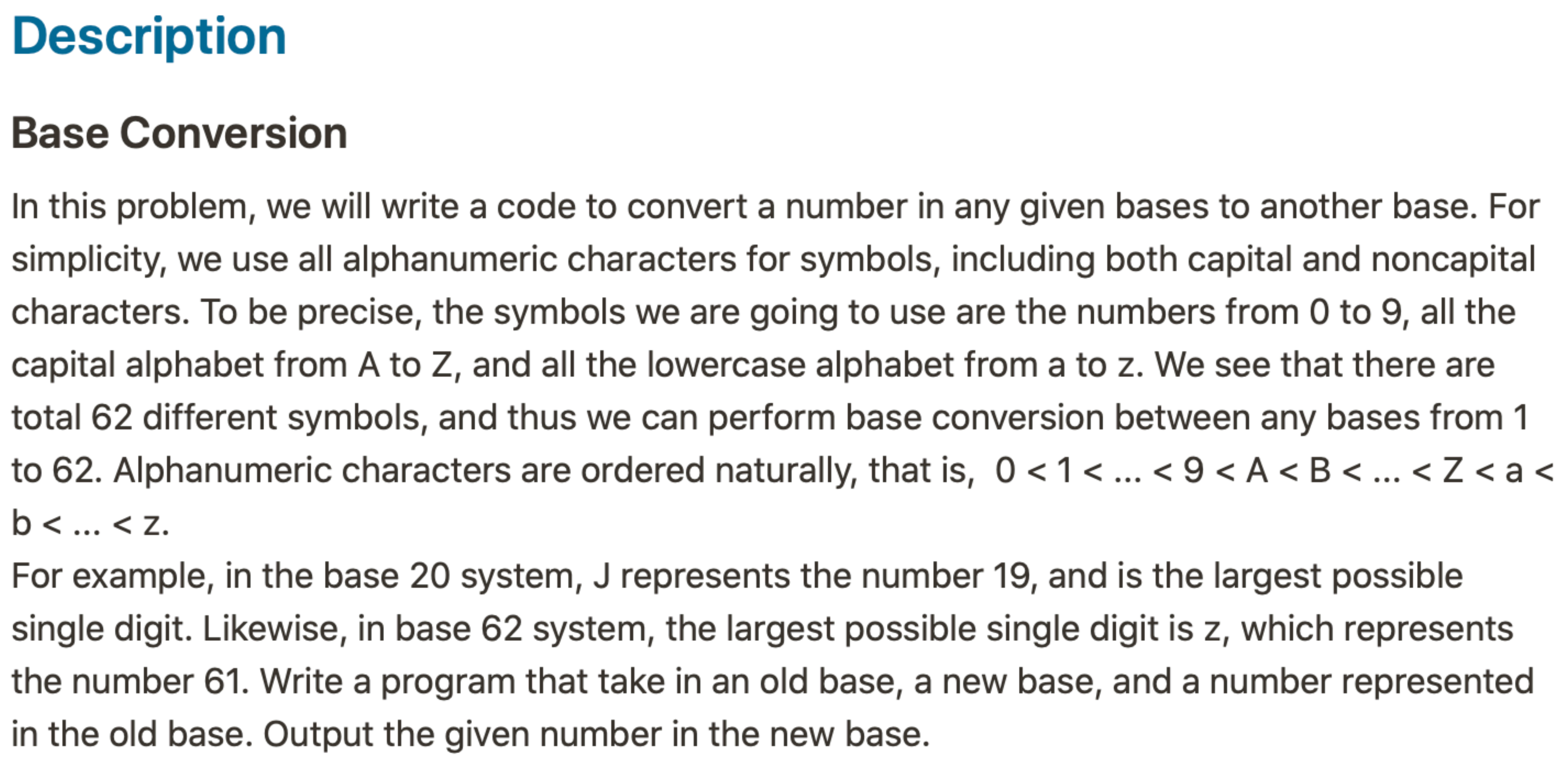


Description

ASCII

ASCII abbreviated from American Standard Code for Information Interchange, is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters. ASCII was indeed originally conceived as a 7-bit code. This was done well before 8-bit bytes became ubiquitous, and even into the 1990s you could find software that assumed it could use the 8th bit of each byte of text for its own purposes ("not 8-bit clean"). Nowadays people think of it as an 8-bit coding in which bytes 0x80 through 0xFF have no defined meaning, but that's a retcon.

Problem 4 - Base Conversion (1.5%)

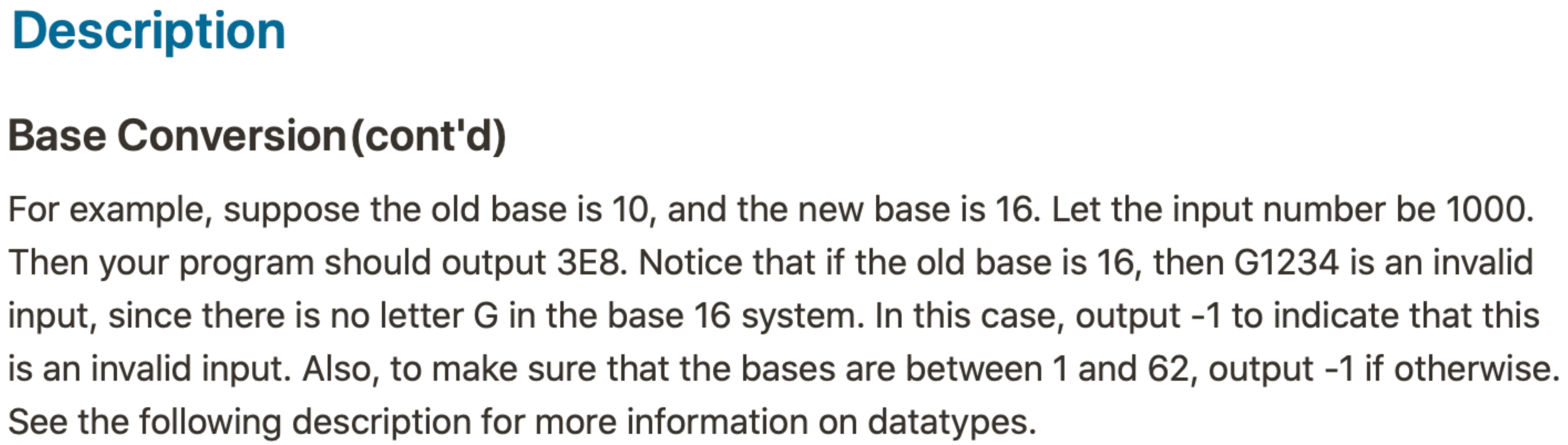


Description

Base Conversion

In this problem, we will write a code to convert a number in any given bases to another base. For simplicity, we use all alphanumeric characters for symbols, including both capital and noncapital characters. To be precise, the symbols we are going to use are the numbers from 0 to 9, all the capital alphabet from A to Z, and all the lowercase alphabet from a to z. We see that there are total 62 different symbols, and thus we can perform base conversion between any bases from 1 to 62. Alphanumeric characters are ordered naturally, that is, 0 < 1 < ... < 9 < A < B < ... < Z < a < b < ... < z.
For example, in the base 20 system, J represents the number 19, and is the largest possible single digit. Likewise, in base 62 system, the largest possible single digit is z, which represents the number 61. Write a program that take in an old base, a new base, and a number represented in the old base. Output the given number in the new base.

Problem 4 - Base Conversion (1.5%)

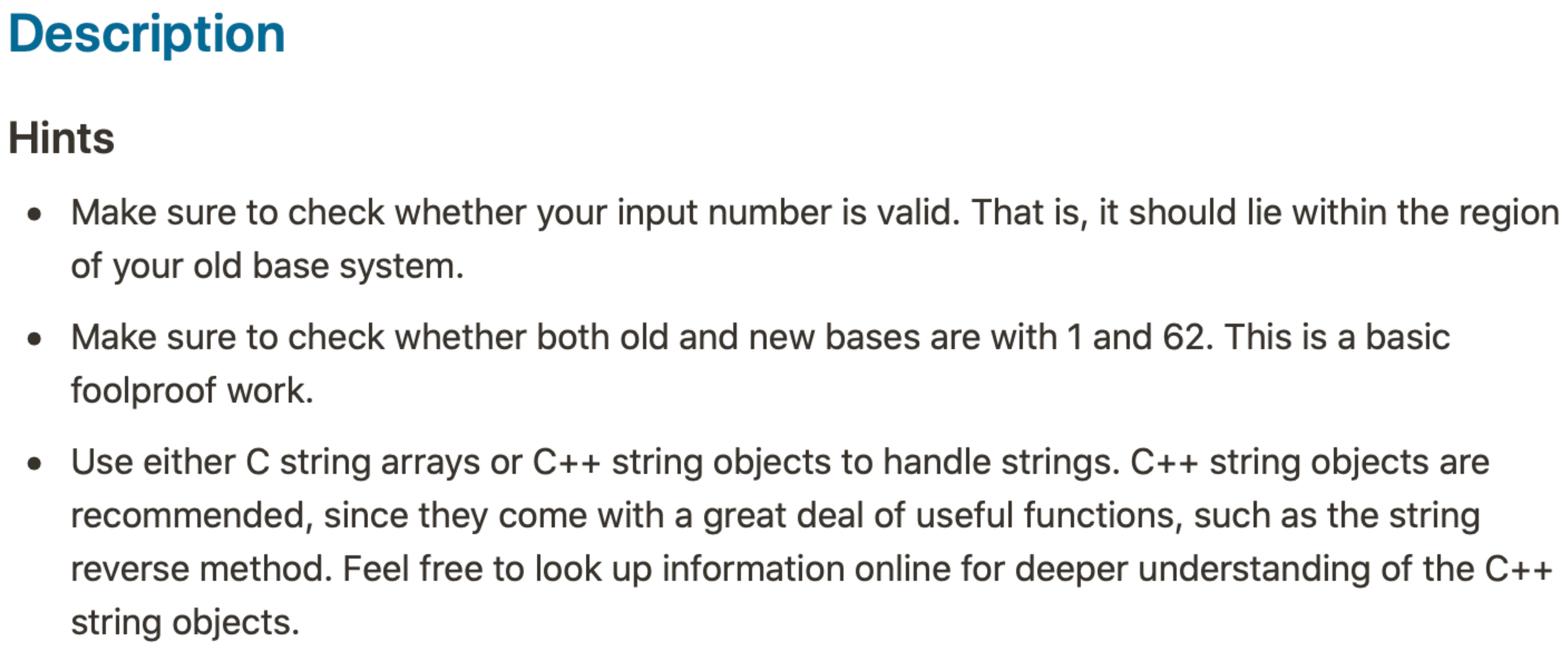


Description

Base Conversion(cont'd)

For example, suppose the old base is 10, and the new base is 16. Let the input number be 1000. Then your program should output 3E8. Notice that if the old base is 16, then G1234 is an invalid input, since there is no letter G in the base 16 system. In this case, output -1 to indicate that this is an invalid input. Also, to make sure that the bases are between 1 and 62, output -1 if otherwise. See the following description for more information on datatypes.

Problem 4 - Base Conversion (1.5%)



Description

Hints

- Make sure to check whether your input number is valid. That is, it should lie within the region of your old base system.
- Make sure to check whether both old and new bases are with 1 and 62. This is a basic foolproof work.
- Use either C string arrays or C++ string objects to handle strings. C++ string objects are recommended, since they come with a great deal of useful functions, such as the string reverse method. Feel free to look up information online for deeper understanding of the C++ string objects.

Problem 4 - Base Conversion (1.5%)

Description

Hints(cont'd)

- Be aware of ASCII values. You should totally understand why `int('a')` is `65`, while `char(65)` is `'a'`. Also, notice that the distribution between numbers and alphabets are not continuous. That is, there is a value gap between the number 9 and the alphabet A.
- Use funtions to modulize your code.
- If you need more sample input, search for key words like "hexidacimal to decimal calculator" for more inputs. Also, try basic base conversion such as binary to decimal and vice versa.
- There will be LOTS of test cases! Be prepared for weird inputs. However, the numbers given won't be unreasonably large, for the sake of the overflow problem in memory.

Problem 4 - Base Conversion (1.5%)

Input

- An old base (`integer` between 1 and 62)
- A new base (`integer` between 1 and 62)
- A number represented in the old base (`string`)

Output

- The given number represented in the new base (`string`)

Problem 4 - Base Conversion (1.5%)

Sample Input

```
30 // base 30 system
16 // hexadecimal
123456 // represented in base 30
```

C++

Sample Output

```
18CCD14 // represented in base 16
```

C++

File name

{Student_ID}_4.cpp

References

1. https://math.libretexts.org/
2. Wikipedia (ASCII)

Problem 5 - Elastic Collision (1.5%)

Description

In this problem, you will be given a rectangular frame, and a point (painted red initially).

Additionally, the starting velocity for the point, and a time interval will also be provided.

The point travels within the frame starting from the bottom-left corner of the rectangular frame, also with its initial starting velocity.

Everytime the point hits an edge or corner of the frame, an elastic collision will occur and the point will change its color to the next one following this sequence :

red → orange → yellow → green → blue → indigo → violet → red → ...

Problem 5 - Elastic Collision (1.5%)

Description

Now, you must write a program to determine which color the point will be after the given time interval.

This problem is also known as the "Bouncing DVD Logo Problem".

Problem 5 - Elastic Collision (1.5%)

Input

w h (a pair of floats representing the width and height of the frame)

vx vy (another pair of floats representing the starting velocity of the point)

t (an integer representing the time interval)

Note :

1. You do not need to consider friction
2. The starting velocity will always aim towards the upper-right direction

Output

the color of the dot (string)

Problem 5 - Elastic Collision (1.5%)

Sample input

```
8.4 6.1
7.5 9.2
12
```

Plain Text

Sample output

```
red
```

Plain Text

File name

{Student_ID}_5.cpp