CMPT365 Programming Assignment 2

Junchen Li

301385486

2021/4/7

**Environment:**

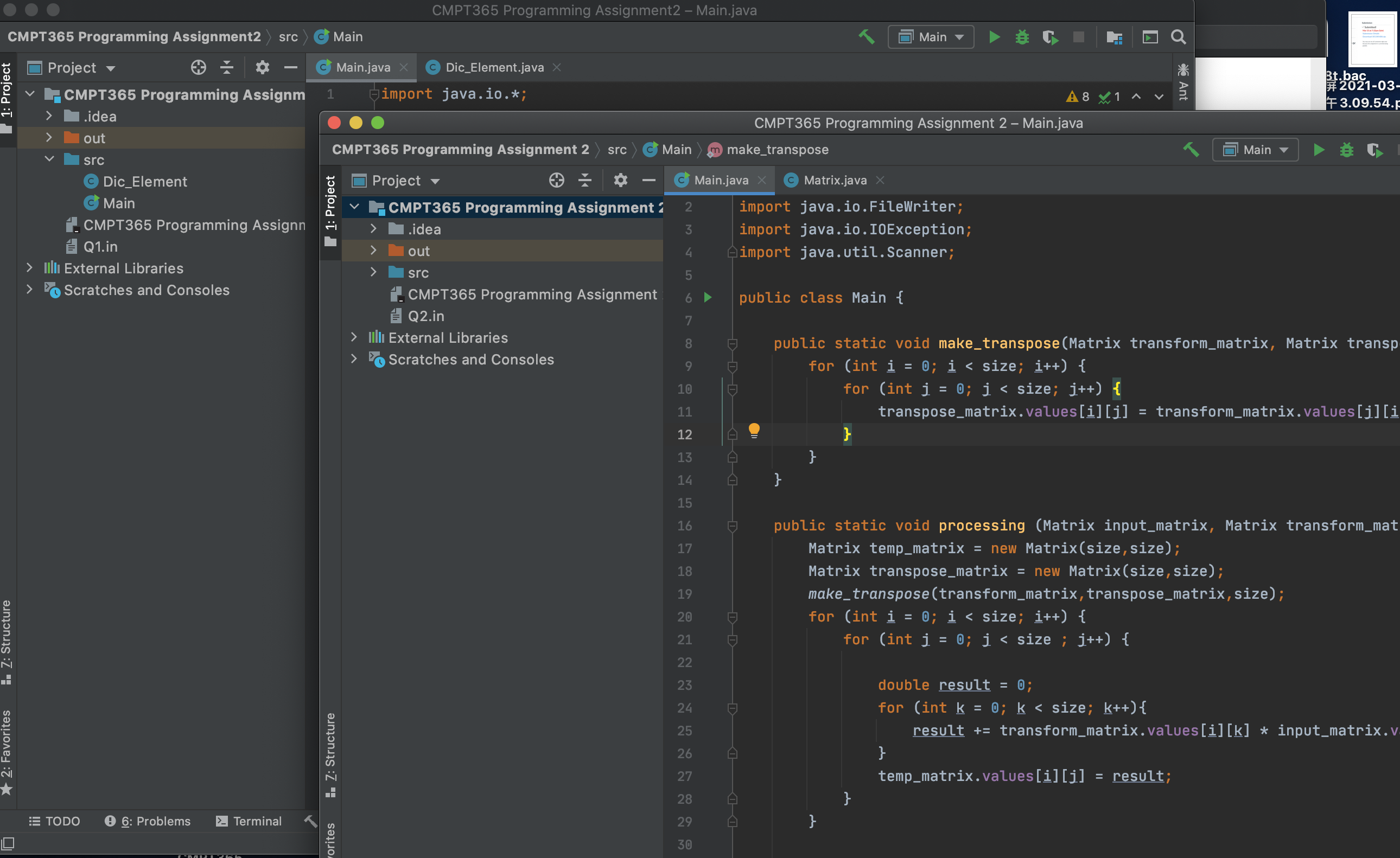
For the second programming assignment, I chose the java as my coding language. Running in the IntelliJ IDEA CE. For the first question I use the libraries as follow:

import java.io.\*; This is for reading the file and writing into the file  
import java.util.ArrayList; Using the array list structure for storing data   
import java.util.Scanner; Scanner is used for reading in the file

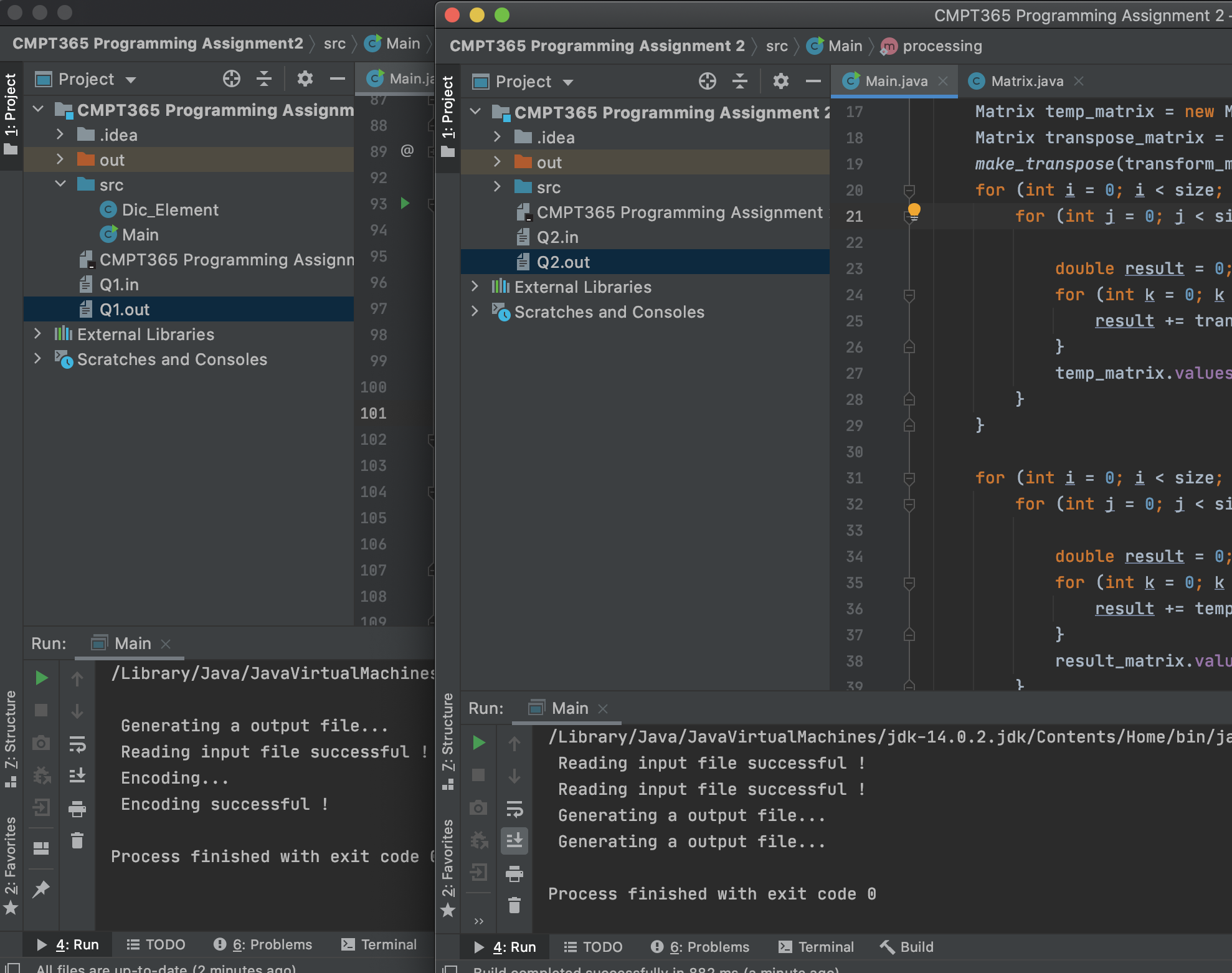
For the question 2, the libraries as follow:

import java.io.FileReader; For reading the file  
import java.io.FileWriter; For writing into the file  
import java.io.IOException; Throw an exception if not found file  
import java.util.Scanner; Scanner is used for reading in the file

Actually the first three lines in the second part are belongs to the java.io.\*. I expand them in order to showing which exactly function I used.

**Operate:**

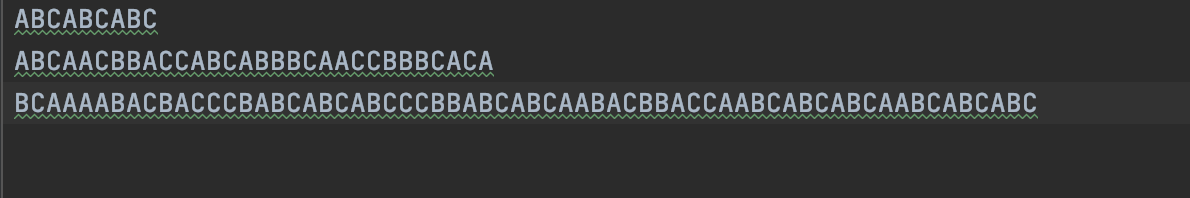
Two blue square show the order of question. Q1.in means the first question and Q2.in means second question in the above screenshot. The only thing is we need to drag and drop the input file in the directory. The current project position directory. If we want to read and write in a specify location. Just adding the absolute path in the code. If we want to running the code, just click on the green arrow in the red box. It will automatically generate two output files for us, and you can see some action statements in the output.



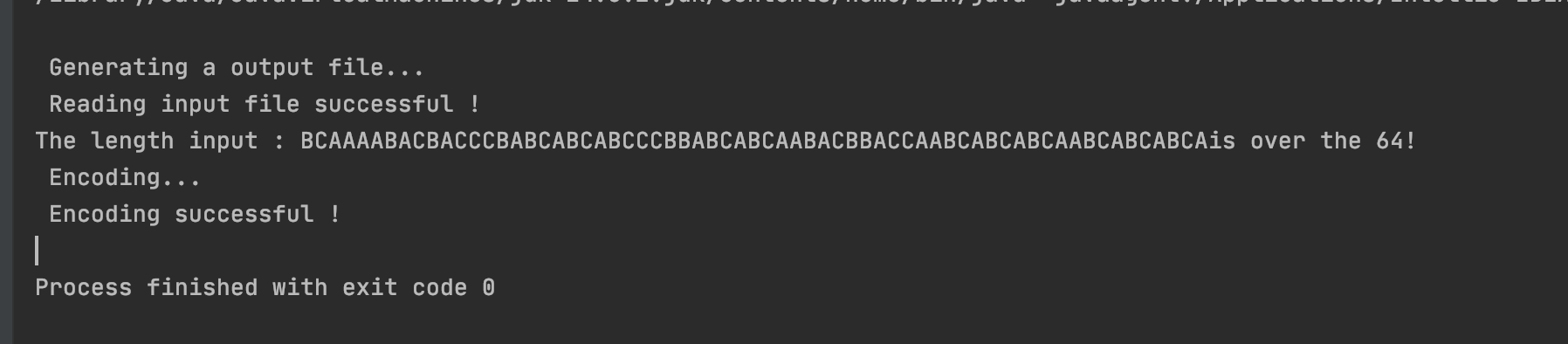
**Resulting:**

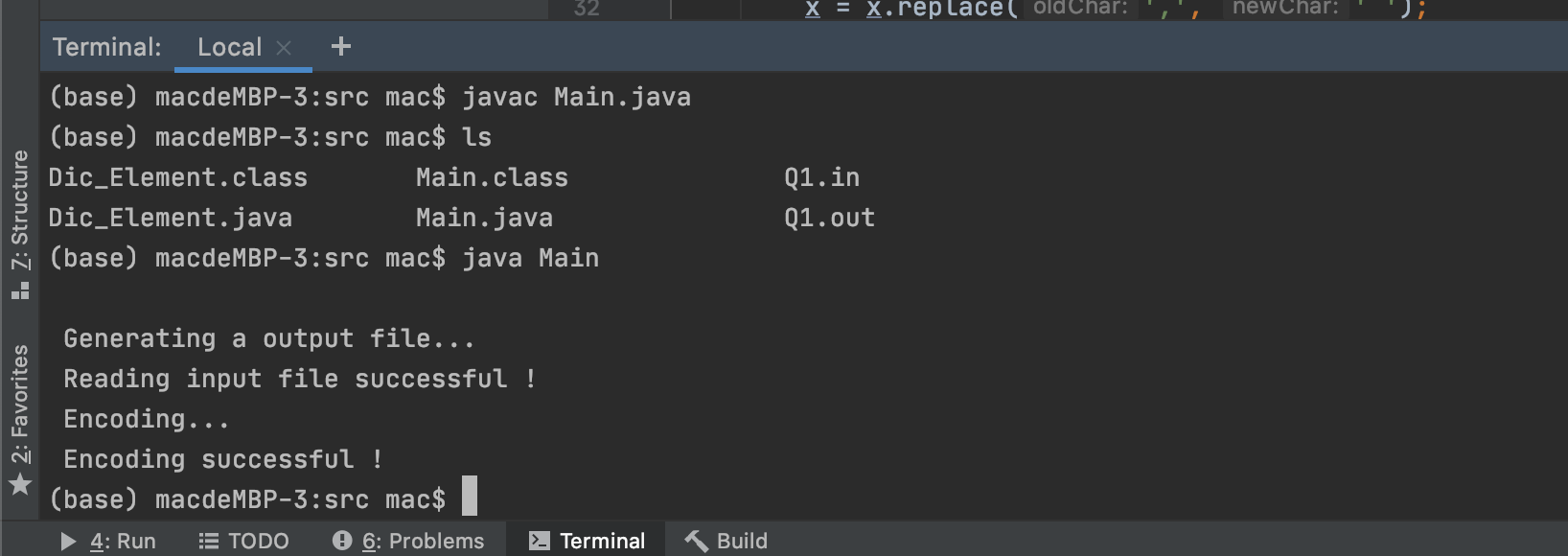
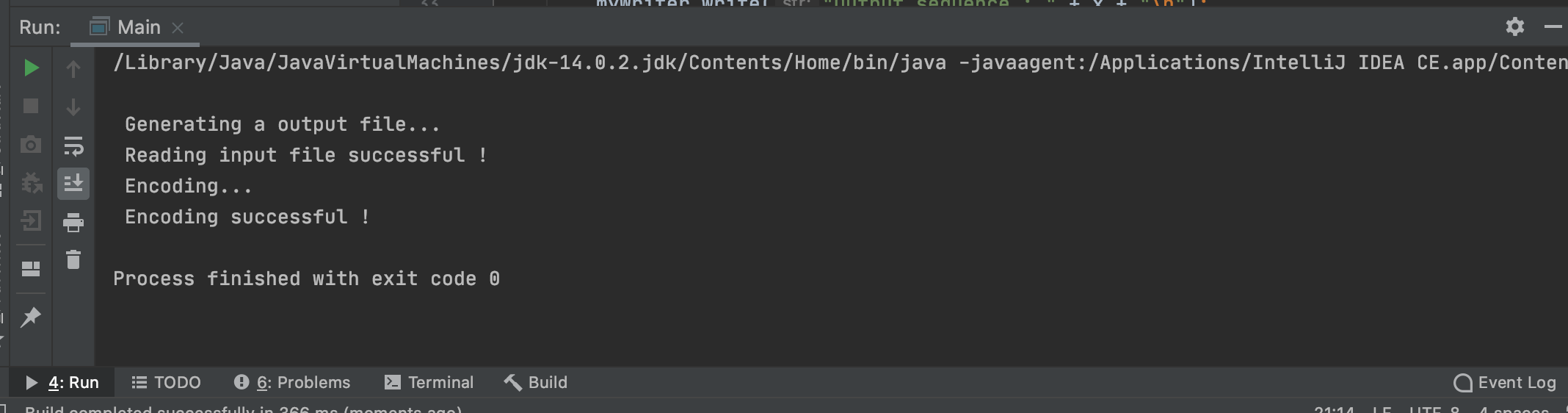
**Q1:**

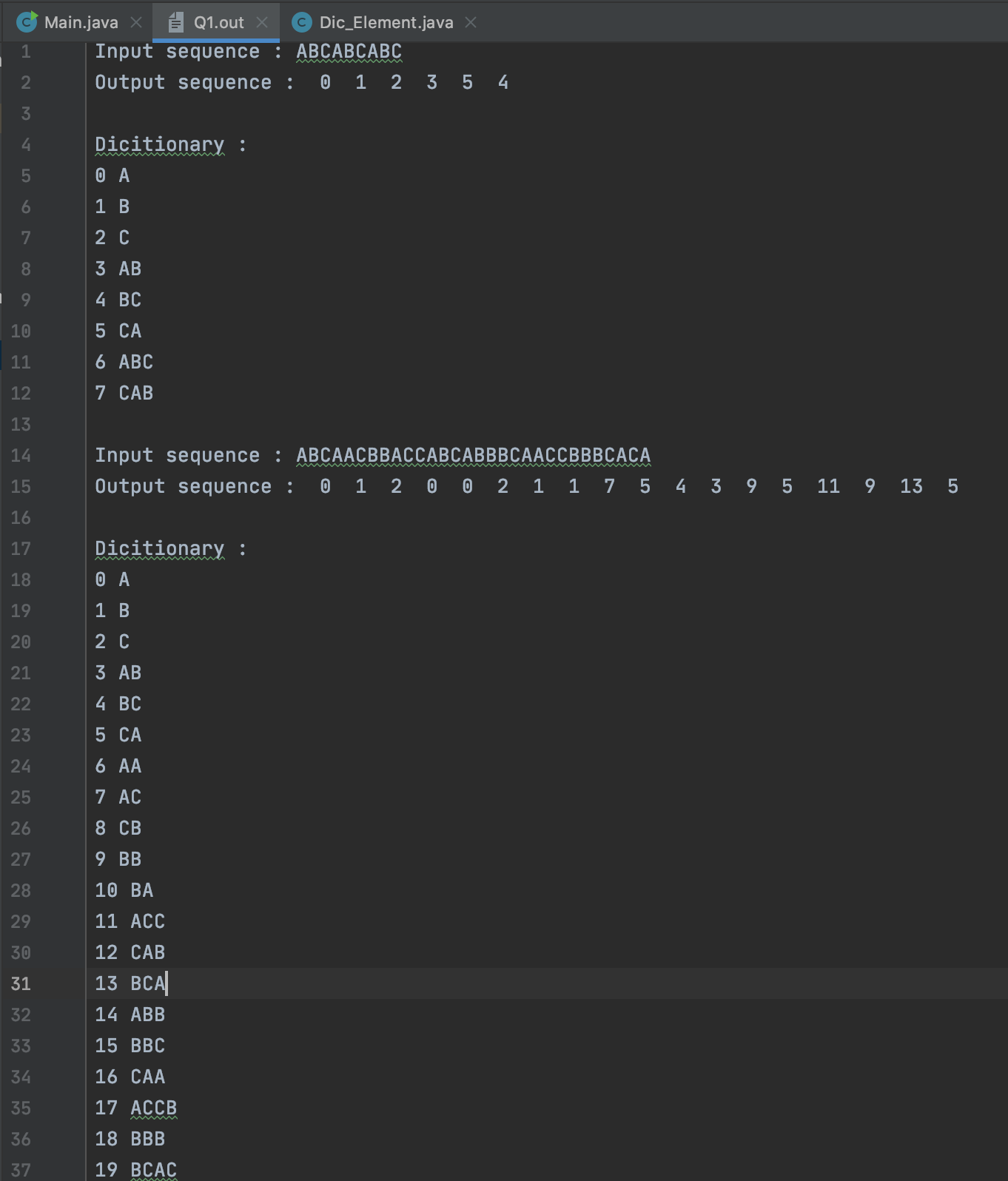
In the first question there are two file, first file is for main function and another one is a class called Dic\_element. I generate three different sequence which their length is in the different range in the Qi.in file.



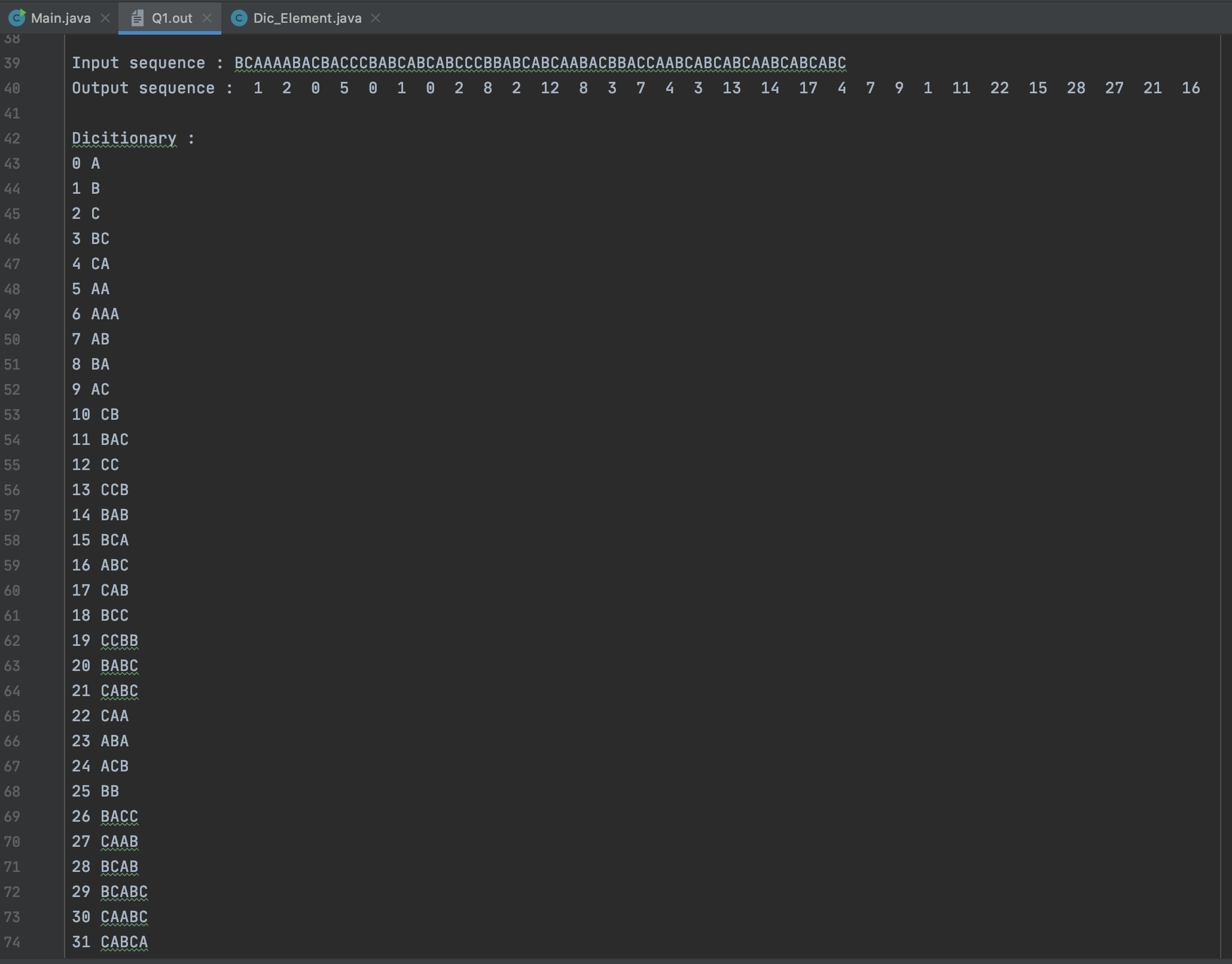
If the length of sequence is over the 64, then it will show the warning message and not count that sequence into the output file.



These two screenshots show the result of compiling and running execution in the terminal and running in the IDE.

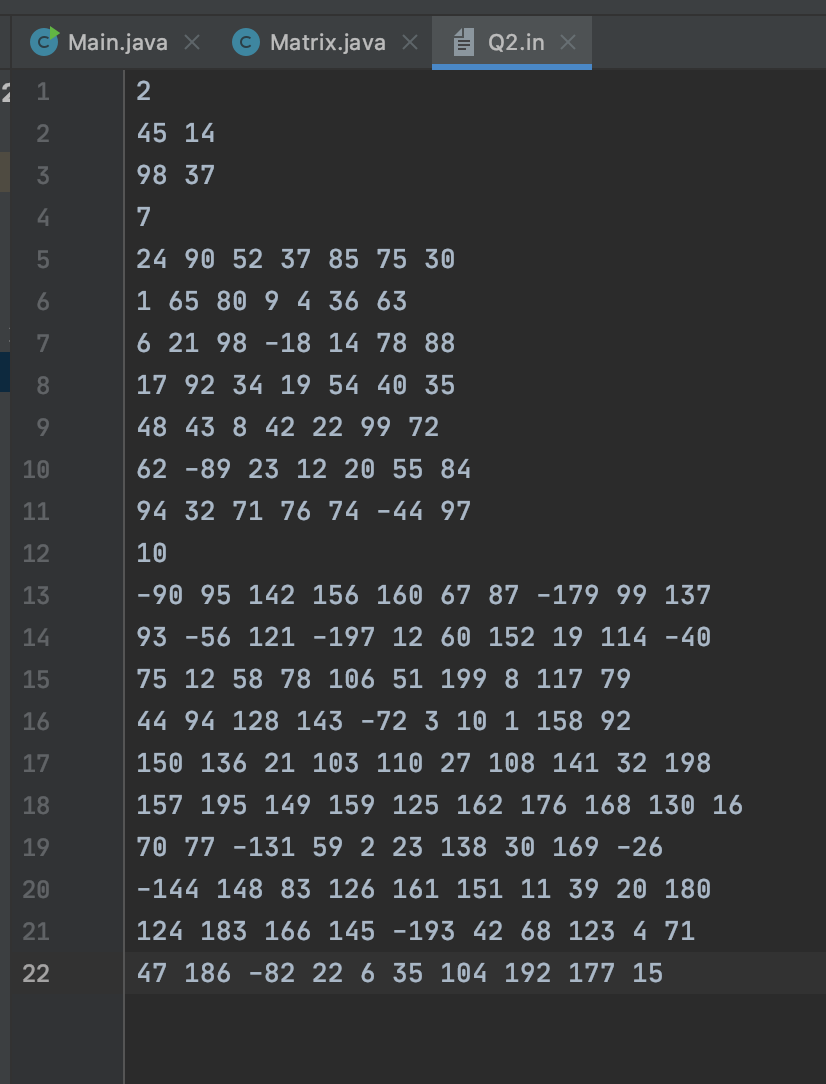
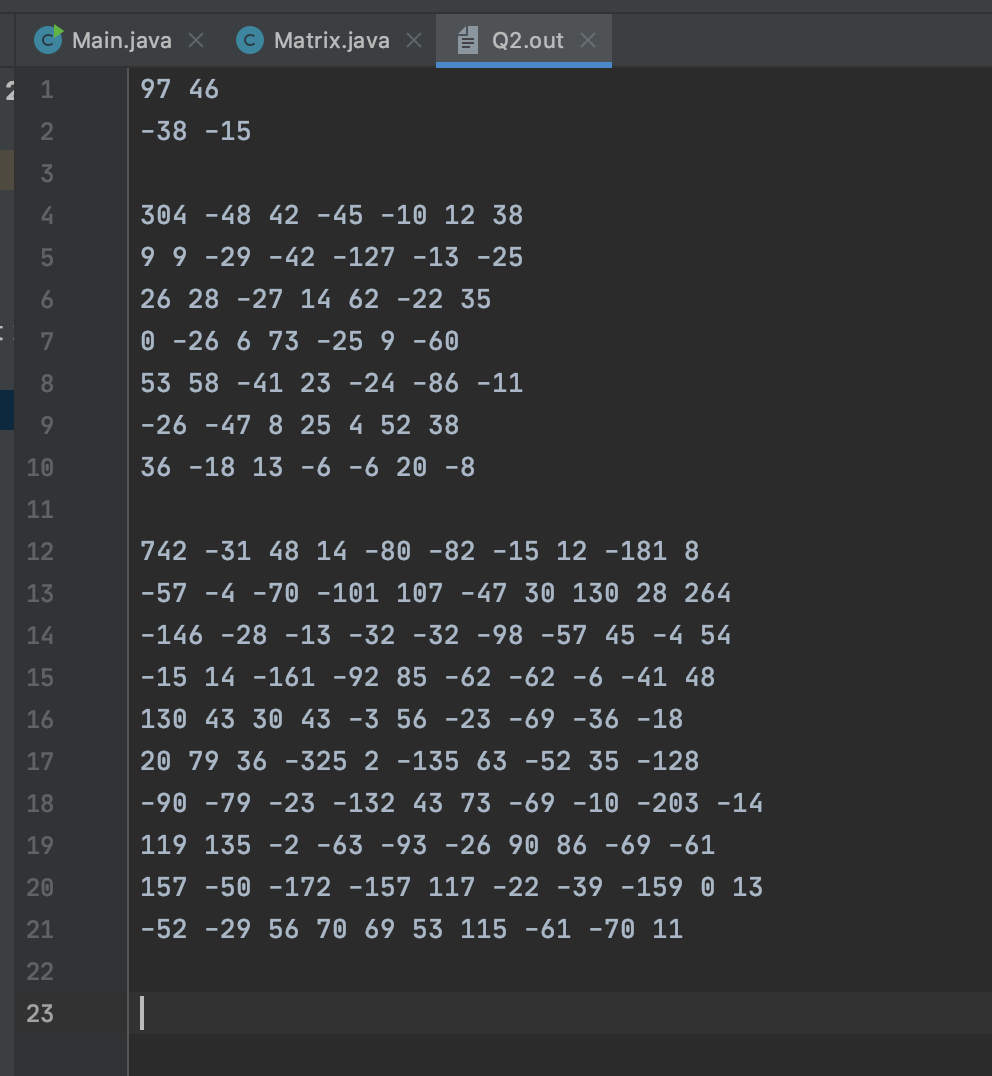
The resulting for the question one as follow:

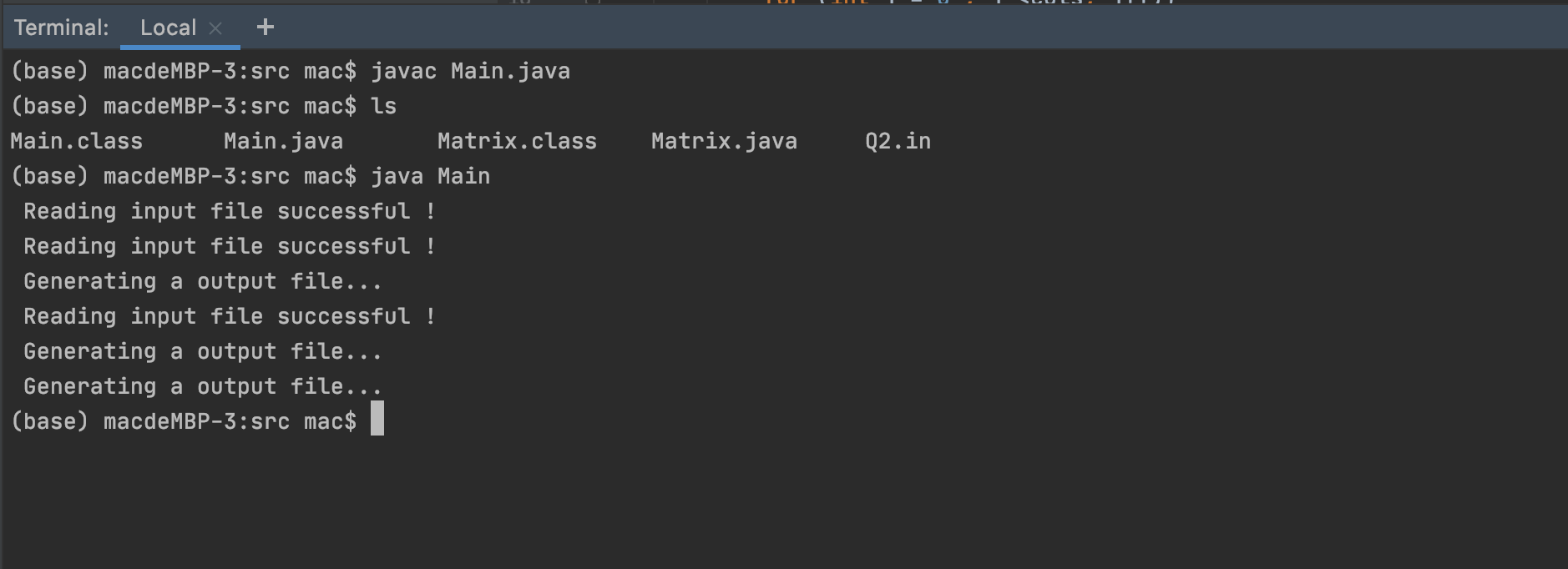
This screenshot contains the result of first two sequence and the dictionary for each one.

This is the last example for the question one. (64 letters)

My code starts in the main function, opens the object file, reads each line and saves them in an array list which has string data type. Generates the initialized dictionary (containing the three known characters and the corresponding numbers). Adapt the pseudocode provided in class and generate the final answer. Since I chose to read all sequences at once, I used a nested loop to read each character. Before proceeding with each question, find the first character (S) and then read the next character (C) in sequence. If some combination of S plus C already exist in the dictionary, let S is equal to S plus C. If not in the dictionary, print the number corresponding to the S character. Add S+C to the dictionary and give it a corresponding number. And so on until you get to the end of the sequence. So now we have a compressed answer. Compression is done three times in the same way. Also I created a new class called DIC\_ELEMENT to act as a dictionary.

**Q2:**

In the second question there are two files, one is for main function an another one is a class called Matrix. In the input file Q2.in, there are different three inputs with different size. Details are shown as follow:

The left one is an input file and right one is output file. They have size two, seven and ten respectively.

For this question, the way of running is as same as the previous one.

These two screenshots show the result of compiling and running execution in the terminal and running in the IDE.

Because the size of the file is different (the first line is usually the size of the matrix, or if there is only one element in a row, that thing is the size of the matrix), the rest of the input matrix is the element. I use two Boolean values to determine whether I need to continue reading or do something later. Once again, when we read a row with only one number, we should do the Discrete Cosine Transform for the previous reading matrix. When we read an input matrix, we need to generate a transformation matrix based on its size. Usually the first row of the transformation matrix has only . N is standard for the size of matrix. Elements after the first row follow the formula . Where i is the row and j is the column, and a = . When we have this matrix. According to formula Y = TX, we also need to find the transpose matrix for the transformation matrix. By definition, rows and columns to be swapped out, we can get marked as . Once again, according to the formula, we're going to multiply three matrices in the order of transformation matrix, input matrix, and transpose matrix together. Then we doing the Discrete Cosine Transform successfully.