Mock Question 1

Write a Java Program that accepts the above **inStrStack** and **inIntQueue** as input parameters and return an **outStrStack** (String Stack) (Top→Bottom) based on the below logic:

- Consider the elements of **inStrStack** from bottom to top and the elements of **inIntQueue** from front to rear.
- For each element in the form "*x:y*" from **inStrStack** (*x* being a String and *y* being a natural number), check if size of *x* is equal to *y*.
    - If equal, check the corresponding element *z* from **inIntQueue** is the square of *y*.
        - If square of *y* is equal to corresponding element *z* from **inIntQueue**, add the element *y* and the element *z* in **outStrStack** in *y:z* format.
        - Otherwise, add *y*:'D' in **outStrStack**.
    - Otherwise add 'D' into **outStrStack**.
- Return the **outStrStack**.

Example:

inStrStack (Top→Bottom): {"Mumbai:8","Mysore:7","Kuala Lumpur:12 ","Tokyo:5","Rome:4"}

inIntQueue (Front→Rear): {16,23,144,49,62}

outStrStack (Top→Bottom): {"D", "D", "12:144","5:D","4:16"}

Sample Input and Output:

inIntStack (Top→Bottom)	inIntQueue (Front→Rear)	outStrStack (Top→Bottom)
{"Delhi: -5","Jodhpur:7"}	{49,25}	{"D","7:D"}
{"Brisbane:8","Sydney:6","Christchurch:10"}	{169,64,64}	{"8:D","6:D","D"}

Mock Question 2

Write a Java program which takes **custNameQueue**(Front->Rear) and **custDobQueue**(Front->Rear)

This program generates coupon code for customers based on following validations.

The **custNameQueue** contains first name and last name of the customers in "FirstName LastName" format. The **custDobQueue** contains date of birth of the corresponding customer in **"DD-MM-YYY"** format. Return the **coupnCodeStack** containing coupon code in format "NameStr-NumCode-DiscountPercent"

- Check if the year of birth lies between any of the following range. The same discount percentage will be provided to the customer.

range	DiscountPercent
2008 – 2012	3
2013 – 2017	20

- If the year of birth does not lie in any of above range, then use 'X' as DiscountPercentage.
- Generate NumCode using the sum of digits of date from date of birth.
- Generate NameStr using first 3 letters of first name in upper case and last 3 letters of last name in lower case.

Example:

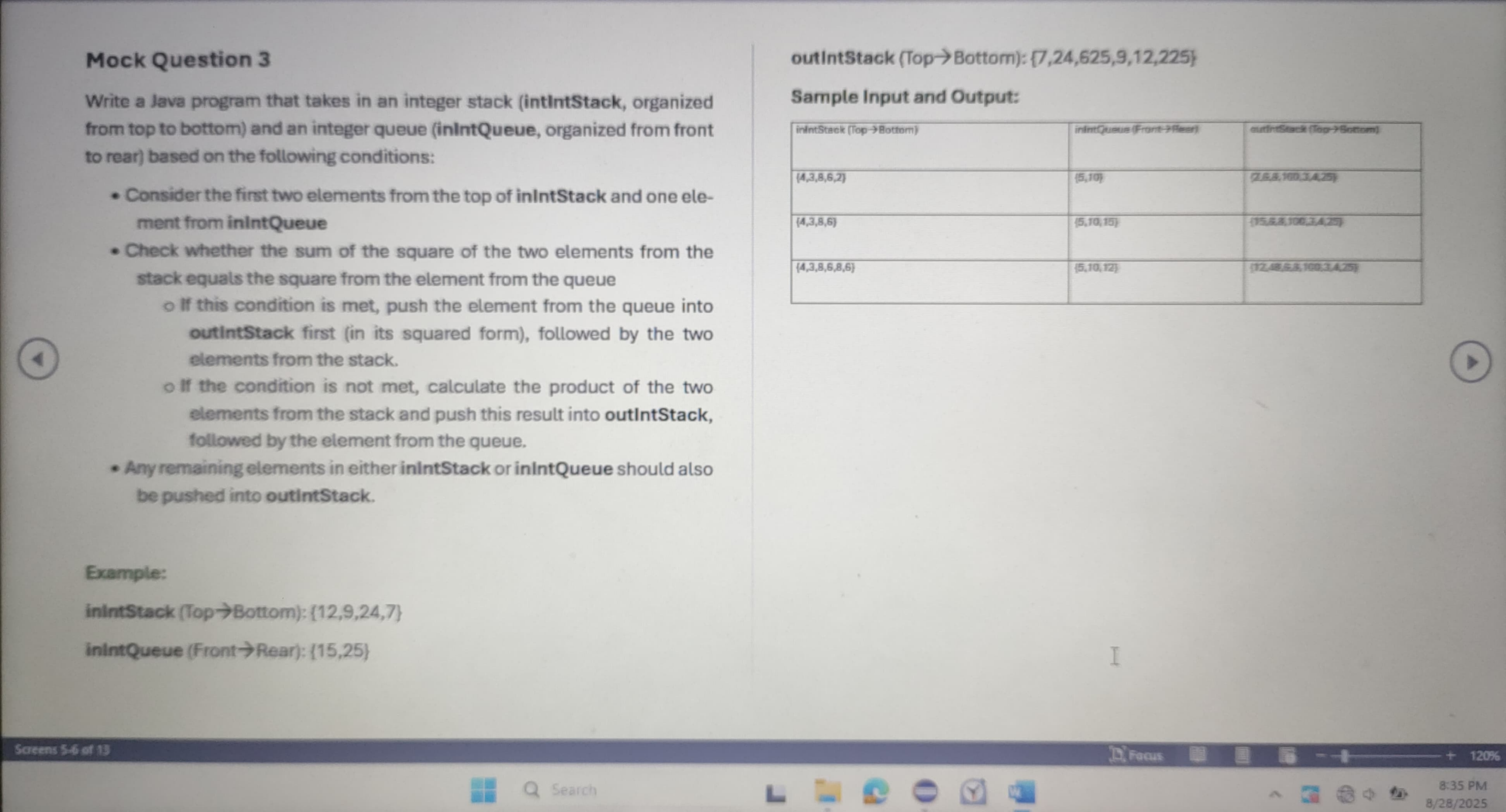
custNameQueue (Front→Rear): {"Paul Steven","Elisa Perry","James Gun","John Deol"}

custDobQueue (Front→Rear): {“12-07-2009”,”20-06-2015”,”31-10-2007”,”19-12-2010”}

coupnCodeStack (Top→Bottom): {"JOHeol-10-35", "JAMgun-4-X", "ELIrry-2-20","PAUven-3-35"}

Sample Input and Output:

custNameQueue (Front→Rear)	**custDobQueue** (Front→Rear)	**coupnCodeStack** (Top→Bottom)
{"Harvey Specter","Donna Paulson", "Michael Ross}	{"27-03-2014",25}	{"D","7:D"}
{"Sheldon Cooper", "Rajesh Koothrapalli"	{169,64,64}	{"8:D","6:D","D"}



Mock Question 3

Write a Java program that takes in an integer stack (**intIntStack**, organized from top to bottom) and an integer queue (**inIntQueue**, organized from front to rear) based on the following conditions:

- Consider the first two elements from the top of **inIntStack** and one element from **inIntQueue**
- Check whether the sum of the square of the two elements from the stack equals the square from the element from the queue
  - If this condition is met, push the element from the queue into **outIntStack** first (in its squared form), followed by the two elements from the stack.
  - If the condition is not met, calculate the product of the two elements from the stack and push this result into **outIntStack**, followed by the element from the queue.
- Any remaining elements in either **inIntStack** or **inIntQueue** should also be pushed into **outIntStack**.

Example:

inIntStack (Top→Bottom): {12,9,24,7}

inIntQueue (Front→Rear): {15,25}

outIntStack (Top→Bottom): {7,24,625,9,12,225}

Sample Input and Output:

inIntStack (Top→Bottom)	inIntQueue (Front→Rear)	outIntStack (Top→Bottom)
{4,3,8,6,2}	{5,10}	{2,6,8,100,3,4,25}
{4,3,8,6}	{5,10,15}	{15,6,8,100,3,4,25}
{4,3,8,6,8,6}	{5,10,12}	{12,48,6,8,100,3,4,25}

Mock Question 4

Write a Java program that takes two input parameters: **intIntStack** and **inIntQueue**. The program should return an output stack, based on the following conditions:

- For each element in inIntStack(from Top to Bottom) and inIntQueue (from Front to Rear):
    - o If the current element from the stack (element1) is greater than the current element from the queue (element2), append element2 to element1.
    - o Otherwise, append element1 to element2.
- The program should return the outIntStack containing the results of these comparisons in the same order as they were processed.

Note: Any remaining elements in either inIntStack or inIntQueue that do not have a corresponding element for comparison should be pushed onto outIntStack.

Example:

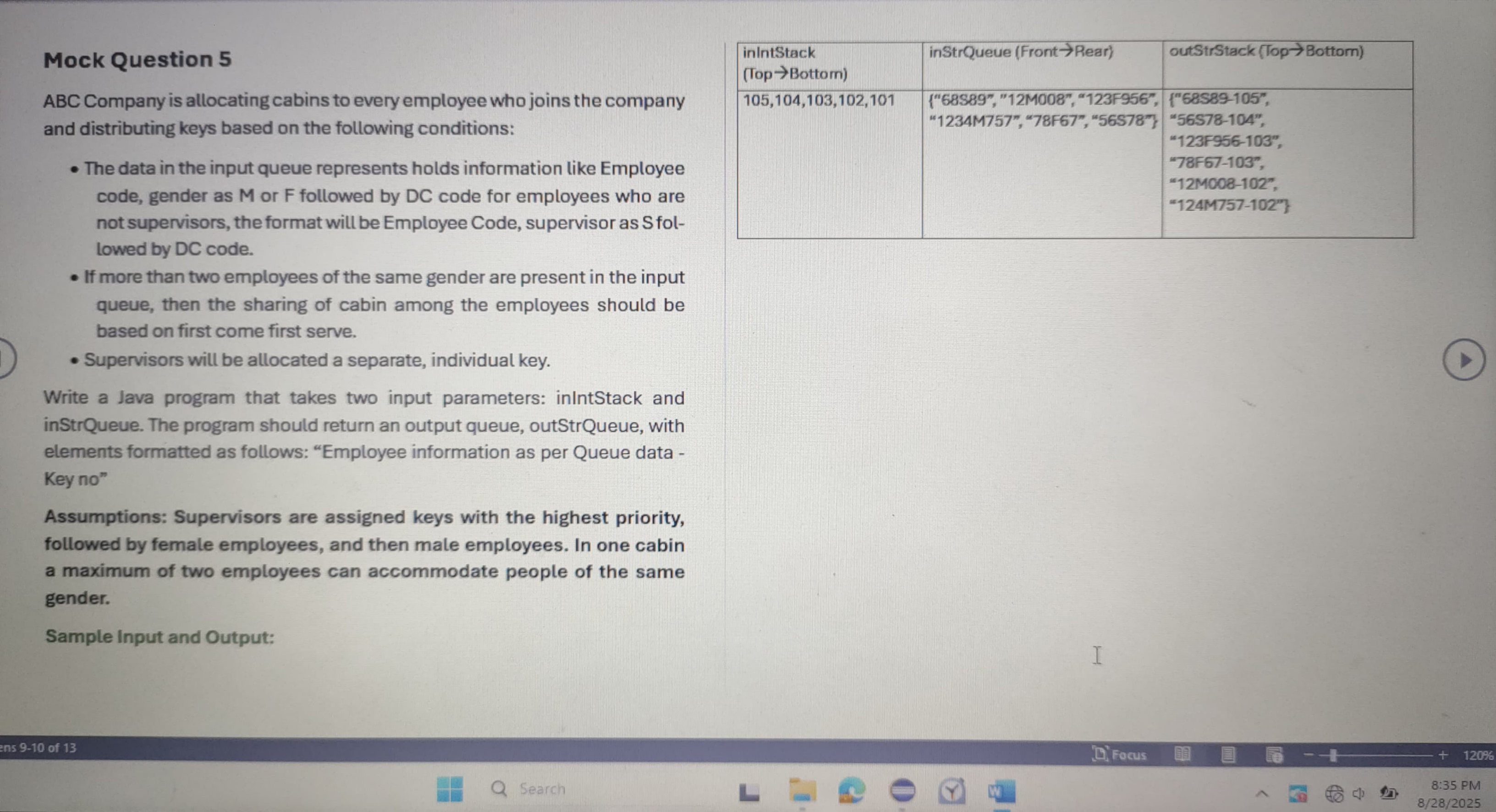
inIntStack (Top→Bottom): {15,20,9,45,5}

inIntQueue (Front→Rear): {20,10,33,35,40}

outIntStack (Top→Bottom): {2015,2010,339,4535,405}

Sample Input and Output:

inIntStack (Top→Bottom)	inIntQueue (Front→Rear)	outIntStack (Top→Bottom)
{6,21,9,18,3}	{27,14,12,33,10,19}	{276, 2114, 129 ,3318, 103, 19}
{10,8,6,4,2,25,7}	{20,5,15,3,40,35}	{2010,85,156,43,402,3525,7}



Mock Question 5

ABC Company is allocating cabins to every employee who joins the company and distributing keys based on the following conditions:

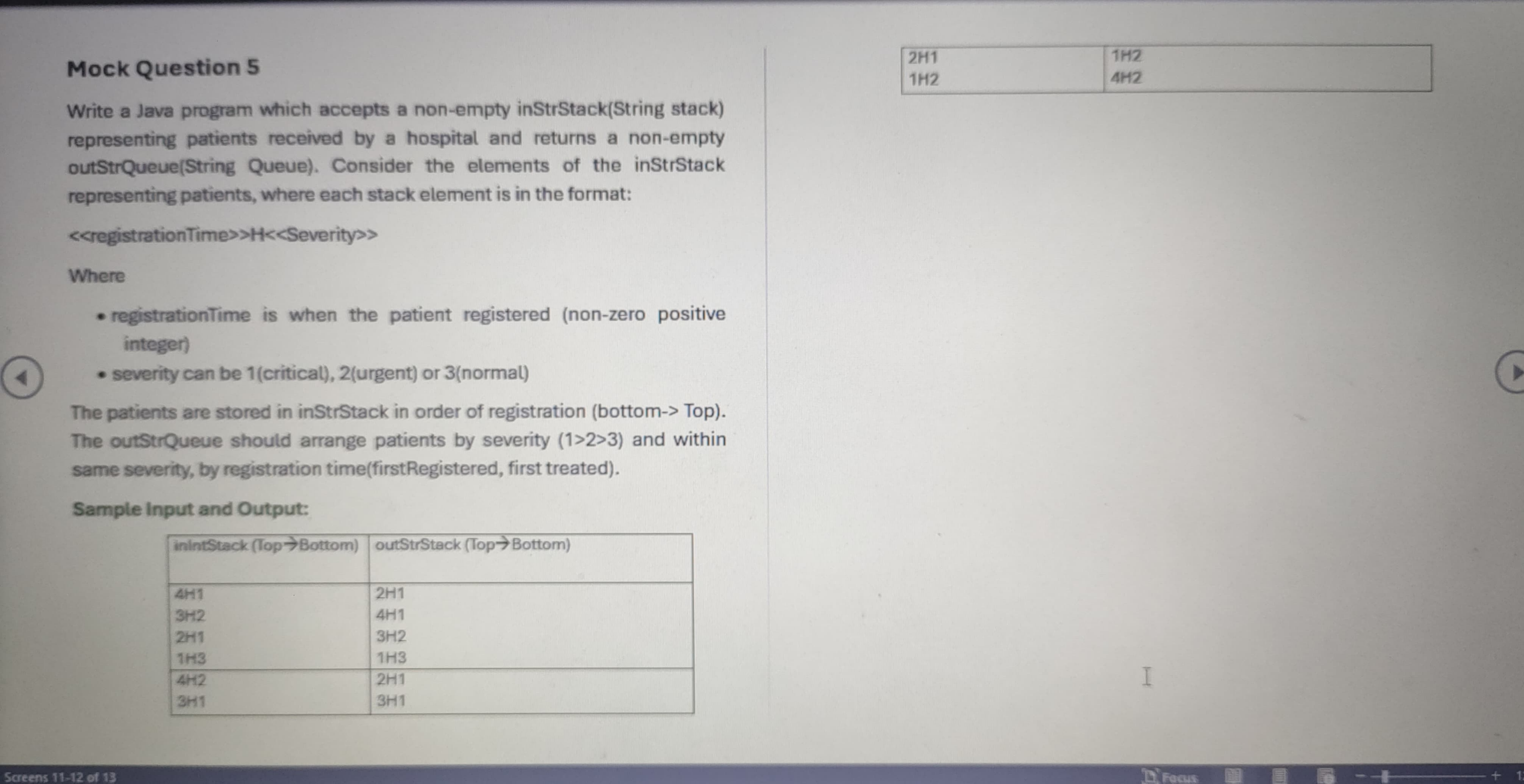
- The data in the input queue represents holds information like Employee code, gender as M or F followed by DC code for employees who are not supervisors, the format will be Employee Code, supervisor as S followed by DC code.
- If more than two employees of the same gender are present in the input queue, then the sharing of cabin among the employees should be based on first come first serve.
- Supervisors will be allocated a separate, individual key.

Write a Java program that takes two input parameters: inIntStack and inStrQueue. The program should return an output queue, outStrQueue, with elements formatted as follows: "Employee information as per Queue data - Key no"

Assumptions: Supervisors are assigned keys with the highest priority, followed by female employees, and then male employees. In one cabin a maximum of two employees can accommodate people of the same gender.

Sample Input and Output:

inIntStack (Top→Bottom)	inStrQueue (Front→Rear)	outStrStack (Top→Bottom)
105,104,103,102,101	{"68S89","12M008","123F956", "1234M757","78F67","56S78"}	{"68S89-105", "56S78-104", "123F956-103", "78F67-103", "12M008-102", "124M757-102"}



Mock Question 5

Write a Java program which accepts a non-empty inStrStack(String stack) representing patients received by a hospital and returns a non-empty outStrQueue(String Queue). Consider the elements of the inStrStack representing patients, where each stack element is in the format:

<<registrationTime>>H<<Severity>>

Where

- registrationTime is when the patient registered (non-zero positive integer)
- severity can be 1(critical), 2(urgent) or 3(normal)

The patients are stored in inStrStack in order of registration (bottom-> Top). The outStrQueue should arrange patients by severity (1>2>3) and within same severity, by registration time(firstRegistered, first treated).

Sample Input and Output:

inIntStack (Top→Bottom)	outStrStack (Top→Bottom)
4H1 3H2 2H1 1H3	2H1 4H1 3H2 1H3
4H2 3H1 2H1 1H2	2H1 3H1 1H2 4H2