1111 物件導向設計與實習(資訊二合)[3670/3671]期中考試

考試說明

- ✓ 考試時間：2022 年 11 月 8 日 18:20~21:40 (最後交卷時間 21:40)。
- ✓ 考試地點：資電 234 電腦教室。(有特殊狀況同學請聯繫老師)
- ✓ 考試題目：公告於 iLearn 2.0 期中考試題目區，可先行思考解題方式。
- ✓ 作答方式：
  - **每題的答案至少須包含程式碼與執行結果，有簡要解題方法說明為佳。請將個人最後作答資料轉成「Mexam_學號.pdf」檔案於最後交卷時間前交到 iLearn 2.0 期中考試答案區。**
  - 考試題目中若有提供程式碼樣板，請盡量以程式碼樣板為基礎作答，但不限於只可用該程式碼樣板作答。**原始作答的使用到的相關程式碼(含註解)、程式輸出入資料畫面等，可以於答案紙中清楚註明參考附件檔名或逕行貼於答案紙中，一併交到 iLearn 2.0 期中考試答案區。**
- ✓ 計分原則：
  - 考試題目每題會標註配分百分比[X%]，視作答正確性與題目要求完整性給予 X 內的分數。
  - 對於特定程式撰寫題目之作答超過題目基本要求並具備作法描述者，該配分得增加 [X%+Y%]中之 Y%加碼，該題以 X+Y 內的分數評分。
  - 本次配分含加碼總計 120 分，總分超過 100 分者最高以 100 分計算。
- ✓ 注意事項：
  - 本次期中考試可利用書籍或使用電腦上網找尋相關資料，但**禁止於考試過程中將先行公告題目之答案以各種具體形式帶入考試地點**。
  - 考試期間請同學將手機關機，禁止同學相互交談，或使用隨身碟、各類通訊設備、線上社群與同學討論。
  - **部分違反上述規則者，以作答總分*60%計算；嚴重抄襲者，以該題作答總分*20%計算。**

敬祝大家考試順利　作答愉快!!!

考試題目

1. **Magic Square - Flip and Turn**

[30%+4%] (計分標準：依正確性與完整度給分 6%~30%，完成所有對角線加總驗證者加碼 4%)

A magic square of degree n, where n is an odd number greater than 3, is an n×n integer square matrix; the element values of this magic square are 1, 2, 3, …, n^2, and the sums of all rows, all columns, the diagonal, and the anti-diagonal are the same, i.e. $(1+n^2)n/2$.

Based on the algorithm for constructing a magic square that we have practiced in class, write a Java program reads in an odd integer n from 3 to 19 and constructs a magic square of degree n. The magic square is horizontally and vertically symmetrical and can also be rotated 90 degrees, 180 degrees or 270 degrees.

That is, a magic square can be flipped and turned by variations of:

- Starting position, *start*: 0 denoting the middle of the top row, 1 denoting the middle of the rightmost column, 2 denoting the middle of the bottom row, 3 denoting the middle of the leftmost column.
- Moving step, *move*: 0 to the upper left, 1 to the upper right, 2 to the lower right, and 3 to the lower left. When moving to the upper left or the upper right, but encountering an occupied cell, then move to the cell below; when moving to the lower left or the lower right, but encountering an occupied cell, then move to the cell above.

Construct a flip-and-turn magic square. Note that the n×n square matrix is a circular structure connected as a torus in the horizontal and vertical directions, that is to say, the two ends of the horizontal and vertical directions are connected, and the magic square matrix is a three-dimensional "ball" (sphere). Therefore, there are n diagonals and n anti-diagonals. Verification of a flip-and-turn magic square should check all rows and all columns; but checks only one of n diagonals and one of n anti-diagonals.

The following are examples of flip-and-turn magic squares, for n=3:

- Enter position of the starting cell at the center of
  (0: top row, 1: right-most column, 2: bottom row, 3: left-most column)
- Set move step (0: upper-left, 1: upper-right, 2: lower-right, 3: lower-left)

start=0 move=0			start=0 move=1			start=0 move=2			start=0 move=3		
6	1	8	8	1	6	8	1	6	6	1	8
7	5	3	3	5	7	4	9	2	2	9	4
2	9	4	4	9	2	3	5	7	7	5	3

start=1 move=0			start=1 move=1			start=1 move=2			start=1 move=3		
4	2	9	2	4	9	7	3	5	3	7	5
8	6	1	6	8	1	6	8	1	8	6	1
3	7	5	7	3	5	2	4	9	4	2	9

start=2 move=0			start=2 move=1			start=2 move=2			start=2 move=3		
7	5	3	3	5	7	4	9	2	2	9	4
2	9	4	4	9	2	3	5	7	7	5	3
6	1	8	8	1	6	8	1	6	6	1	8

start=3 move=0			start=3 move=1			start=3 move=2			start=3 move=3		
9	4	2	9	2	4	5	7	3	5	3	7
1	8	6	1	6	8	1	6	8	1	8	6
5	3	7	5	7	3	9	2	4	9	4	2

You may start with program skeleton **Mexam_skeleton_1.java** and change the file name to **Mexam_Dxxxxxxx_1.java** where **Dxxxxxxx** is your student ID. The basic output examples of this program as follows.

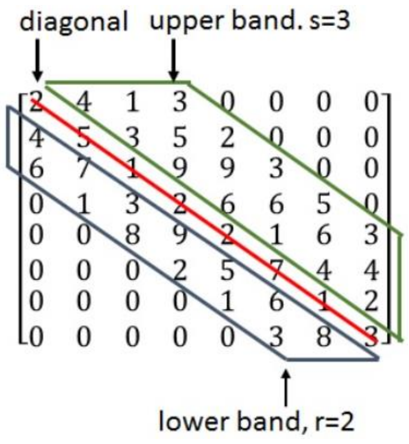
```
Enter the size of the magic square (3<=n<20, odd integer): 5
Enter position of the starting cell at the center of
  0: top row, 1: right-most column, 2: bottom row, 3: left-most column: 0
Set move step (0: upper-left, 1: upper-right, 2: lower-right, 3: lower-left): 0
The magic square of degree 5:
        15   8   1  24  17
        16  14   7   5  23
        22  20  13   6   4
         3  21  19  12  10
         9   2  25  18  11
The magic square passes verification.
The sum of each row and each column is 65.
The sum of at least one diagonal is 65.
The sum of at least one anti-diagonal is 65.

Enter the size of the magic square (3<=n<20, odd integer): 5
Enter position of the starting cell at the center of
  0: top row, 1: right-most column, 2: bottom row, 3: left-most column: 2
Set move step (0: upper-left, 1: upper-right, 2: lower-right, 3: lower-left): 1
The magic square of degree 5:
        23   5   7  14  16
         4   6  13  20  22
        10  12  19  21   3
        11  18  25   2   9
        17  24   1   8  15
The magic square passes verification.
The sum of each row and each column is 65.
The sum of at least one diagonal is 65.
The sum of at least one anti-diagonal is 65.
```

2. Banded Matrix

[20%+4%] (計分標準：依答案完整度與正確性給分 4%~20%，矩陣類別封裝者至多加碼 4%)

Base on the Matrix Class practice, we have defined a matric class and written some methods of arithmetic. A kind of sparse matrix is banded matrix. If square matrix A is of size n×n, a lower band element of bandwidth r is the element ai,j such that 0<i-j≤r and an upper band element of bandwidth s is the element ai,j such that 0<j-i≤s. Only the elements on the diagonal, on the lower band, and on the upper band can be non-zero; all other elements are called off-band elements and they are all zeros. The following is an example of an 8×8 banded matrix with the lower bandwidth r of 2 and the upper bandwidth s of 3.



Let the size of square matrix A, B, and C be n×n. Also, let ra and sa be the lower and upper bandwidth of square matrix A, respectively, and rb and sb be the lower and upper bandwidth of square matrix B, respectively. If C=A×B, then the lower bandwidth of C is ra+rb and the upper bandwidth of C is sa+sb with the limit of upper bound n-1. The non-zero elements of $c_{i,j}$ is computed as the following formula:

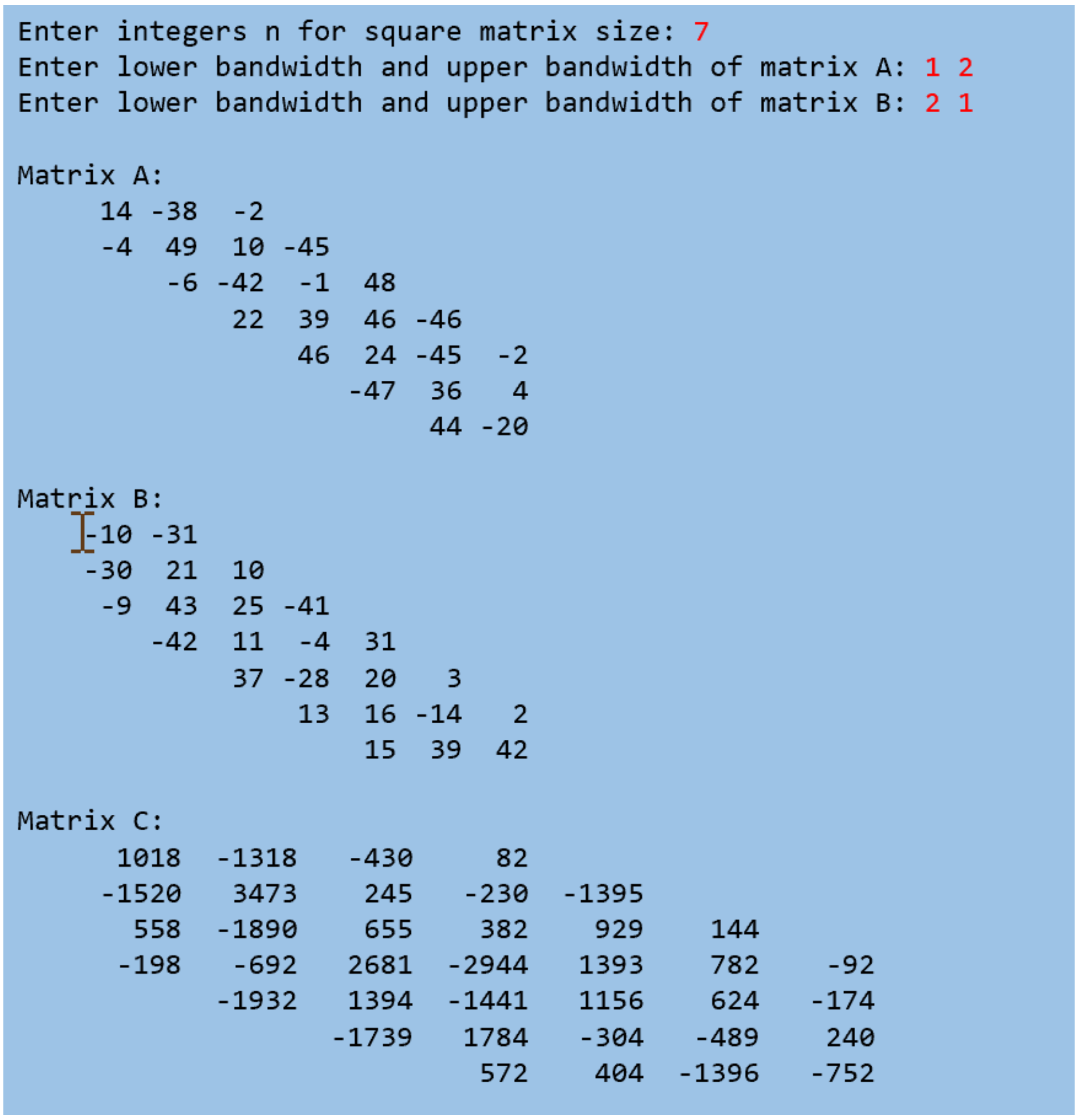
$$\forall i,j: 0 \le i \le n-1 \wedge \max(0, i-ra-rb) \le j \le \min(n-1, i+sa+sb), c_{i,j} = \sum_{k=\max(0,\max(i-ra,j-sb))}^{\min(n-1,\min(i+sa,j+rb))} a_{i,k} \times b_{k,j}$$

In Java package,

- An application class BandedMatrixAPP.Java to input an integer n of square matrix size for matrices A, B, and C, and two pairs of integer, ra and sa as the lower and upper bandwidth of matrix A, and rb and sb as the lower and upper bandwidth of matrix B. Then, compute matrix multiplication C=A×B. Output matrices A, B,and C; do not output the off-band elements form matrices A, B, and C.

- An encapsulated class BandedMatrix.Java to define the arithmetic operations and print matrix method. It can create exact number of matrix elements for banded matrices A, B, and C and randomly generate non-zero ai,j and bi,j for matrices A and B also. Assuming the values of the matrix elements are between -50 and 50, use a random number generator to generate the values of the matrix elements.

You may start with program skeleton **Mexam_skeleton_2.java** and change the file name to **Mexam_Dxxxxxxx_2.java** where **Dxxxxxxx** is your student ID. Separate the application to **Mexam_Dxxxxxxx_2APP.java** and class definition **Mexam_Dxxxxxxx_2Class.java** is also accepted and preferred. The basic output examples of this program as follows.



```
Enter integers n for square matrix size: 7
Enter lower bandwidth and upper bandwidth of matrix A: 1 2
Enter lower bandwidth and upper bandwidth of matrix B: 2 1

Matrix A:
     14 -38  -2
     -4  49  10 -45
         -6 -42  -1  48
             22  39  46 -46
                 46  24 -45  -2
                    -47  36   4
                         44 -20

Matrix B:
    -10 -31
    -30  21  10
     -9  43  25 -41
        -42  11  -4  31
             37 -28  20   3
                 13  16 -14   2
                     15  39  42

Matrix C:
      1018  -1318   -430     82
     -1520   3473    245   -230  -1395
       558  -1890    655    382    929    144
      -198   -692   2681  -2944   1393    782    -92
            -1932   1394  -1441   1156    624   -174
                   -1739   1784   -304   -489    240
                           572    404  -1396   -752
```

3. BigHexInteger Class

[30%+12%] (計分標準:依正確性與完整度給分 6%~30%,完成乘除法與運算頗析至多加碼 12%)

BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. (https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html)

We have written a **Mexam_Dxxxxxxx_3Class.java** program to define a class to inherit the BigInteger class and handle the add operation of six big numbers. Besides the add operation, you must implement subtract operation at least. You can append any methods to the BigHexInteger class if necessary. The multiply and divide operations are preferred.

Write a Java program **Mexam_Dxxxxxxx_3APP.java** to test the arithmetic operations (add and subtract, at least) of BigIneger and BigHexInteger objects. This program declares BigInteger[] array as a container to refer the objects and calculates the result of arithmetic operations of the BigInteger and BigHexInteger objects. The BigInteger[] contains six objects at least. The minus number must be taken in consideration also.

In the application class, get the six big numbers a (BigHexInteger), b (BigInteger), c (BigHexInteger), d (BigInteger), e (BigHexInteger), and f (BigInteger), and print them in the main program.

Calculate and output the following arithmetic expressions at least:

- a + b + c + d + e +f,
- b – c + d + e – f + a

You can implement simple parsing method to parse the above expressions if necessary.

The basic output examples of this program as follows.

```
The testing data as follows:
================================
a = 121 (BigInteger)
b = FF  (BigHexInteger)
c = -28 (BigInteger)
d = -1C (BigHexInteger)
e = 83  (BigInteger)
f = 2B  (BigHexInteger)


The output data as follows:
================================

a+b+c+d+e+f = 446(BigInteger) 1BE(BigHexInteger)

b-c+d+e-f+a = 416(BigInteger) 1A0(BigHexInteger)
```

4. Triathlon

[20%] (計分標準：依正確性與完整度給分 4%~20%)

A triathlon is an endurance multisport race consisting of swimming, cycling, and running over various distances. Triathletes compete for fastest overall completion time, racing each segment sequentially with the time transitioning between the disciplines included.

Write a Java program **Mexam_Dxxxxxxx_4APP.java** to sort the score rank in the class using *Array.sort.* There are three data fields in Triathlon Score class in **Mexam_Dxxxxxxx_4Class.java**. They are Swimming score, Biking score, and Running score respectively. The interval of two transition zones are included and ignored in this program.

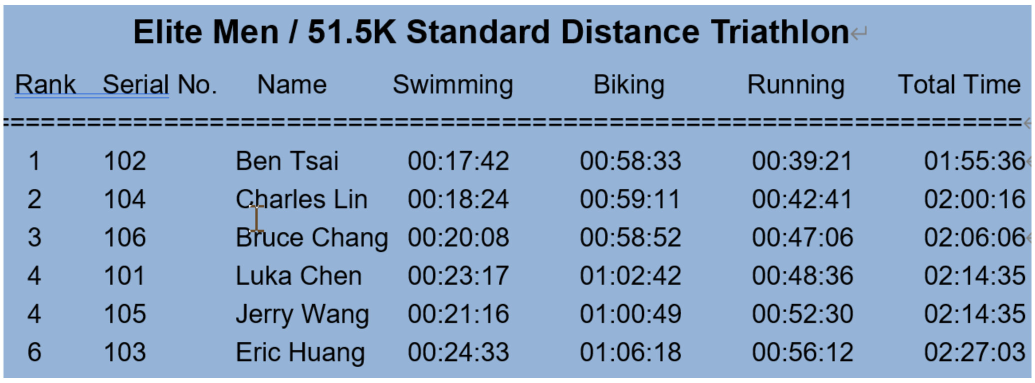
The rank priority is sorted by least time first that is sorted by the total score (the sum of Swimming score, Biking score, and Running score). If there are more than two players with the same score, they will be sorted by serial number but with the same ranking.

Examples of test data are adopted from a standard game as follows, but not limited to this. The score format is hour : minute : second (hh:mm:ss).

SerialNo.	Name	Swimming	Biking	Running
101	Luka Chen	00:23:17	01:02:42	00:48:36
102	Ben Tsai	00:17:42	00:58:33	00:39:21
103	Eric Huang	00:24:33	01:06:18	00:56:12
104	Charles Lin	00:18:24	00:59:11	00:42:41
105	Jerry Wang	00:21:16	01:00:49	00:52:30
106	Bruce Chang	00:20:08	00:58:52	00:47:06

Please implement the *Comparable interface* to handle this sorting problem.

The basic output examples of this program as follows.



```
           Elite Men / 51.5K Standard Distance Triathlon
Rank  Serial No.   Name         Swimming    Biking      Running     Total Time
==============================================================================
  1     102        Ben Tsai     00:17:42    00:58:33    00:39:21    01:55:36
  2     104        Charles Lin  00:18:24    00:59:11    00:42:41    02:00:16
  3     106        Bruce Chang  00:20:08    00:58:52    00:47:06    02:06:06
  4     101        Luka Chen    00:23:17    01:02:42    00:48:36    02:14:35
  4     105        Jerry Wang   00:21:16    01:00:49    00:52:30    02:14:35
  6     103        Eric Huang   00:24:33    01:06:18    00:56:12    02:27:03
```

-- The End --