<u>אלגוריתמים 1</u> סמסטר א' תש"פ

תכנות אלגוריתם לחיפוש מסלולים אופטימאליים בין נקודה (0,0) לנקודה (M,N)

<u>הנחיות כלליות:</u>

- שפת תכנות – JAVA
- הפונקציות שעליכם לכתוב צריכות להיות **יעילות ככל האפשר** (וכמובן נכונות).
- תרגיל זה נעשה <u>ביחידים בלבד</u>.
- מועד אחרון להגשת עבודה **03.02.2020 – 23:55**
- תרגיל זה ייבדק בצורה אוטומטית ע"י תכנית מחשב שתשתמש בשמות המוזכרות להלן.
- את התרגיל יש להגיש לאתר של מידע אישי (קורס אלגוריתמים 1, תחרות).

<u>שם הקובץ – מספר תעודת זהות, סוג הקובץ - rar או zip.</u>

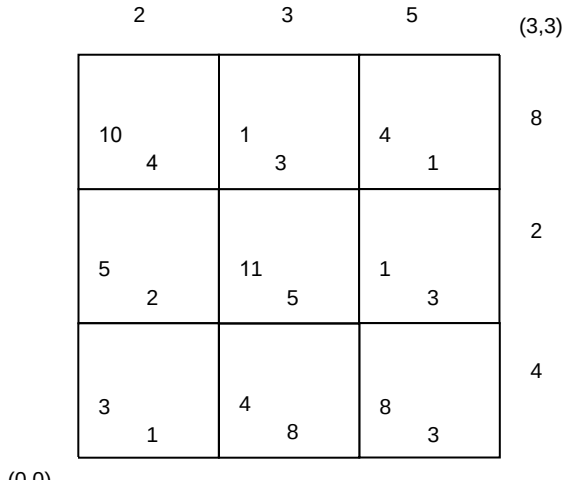
הקובץ שלא יעמוד בדרישות אלו לא ייבדק!

<u>ניסוח הבעיה:</u>

נתונה רשת $M \times N$ עם M קדקודים על ציר ה-X ו-N קדקודים על ציר ה-Y. על כל צלע רשומה עלות מעבר הצלע.
במילים אחרות הקלט הוא מערך דו-ממדי (מטריצה) של קדקודים, כל קדקוד (**Node**) מכיל את העלויות של שתי הצלעות היוצאות ממנו לכיוונים ימינה ומעלה:

x - עלות של הצלע האופקי, (הכיוון, כמו בציר ה-X, ימינה)
y - עלות של הצלע האנכי, (כיוון כלפי מעלה, כמו בציר ה-Y)

<u>דוגמה</u>: במטריצה שלהלן:
קדקוד (0,0) נראה כך: $x = 1; y = 3;$
קדקוד (0,3) נראה כך: $x = 0; y = 4;$
קדקוד (3,3) נראה כך: $x = 0; y = 0;$



המשימה מתחלקת לשני חלקים. בחלק א' יש להתייחס למסלולים בעלי עלות קטנה ביותר.
בחלק ב' יש להתייחס למסלולים בעלי עלות הבאה בגודלה, נקרא את העלות הזו "*עלות משנית*". במילים אחרות עלות משנית היא עלות מינימאלית הגדולה מעלות קטנה ביותר.

חלק א' מסלולים בעלי עלות קטנה ביותר

בחלק זה עליכם למצוא את מספר מסלולים קצרים ביותר ומספר מסלולים אופטימאליים בין נקודה (0,0) לנקודה (M,N). עליכם גם לבנות את כל המסלולים הקצרים ביותר ואת כל המסלולים האופטימאליים כאשר מספר המסלולים הקצרים ביותר לא עולה על מספר נתון **teta**.

מסלול קצר ביותר הוא מסלול בעל עלות מינימאלית.

מסלול אופטימאלי הוא מסלול בעל עלות מינימאלית שמספר הפניות (פניה = שינוי כיוון התנועה) הוא קטן ביותר.

הנחיות לתכנות:

1. מבנה של קדקוד אחד מיוצג ע"י מחלקת **Node**:

```
public class Node{
      double y, x;
      public Node(double x, double y){
            this.x = x;
            this.y = y;
            . . . . . .
      }
}
```

ניתן להוסיף למחלקת Node משתני עצם נוספים לפי הצורך.

2. כתבו מחלקה בשם **BestPath** לחישוב המסלולים האופטימאליים בין נקודה (0,0) לנקודה (M,N). עליכם להגדיר משתני עצם ומתודות לפי הצורך. המחלקה חייבת להכיל מתודות הבאות וכמובן, מתודות עזר לפי הצורך.

2.1. בנאי המחלקה: **BestPath(Node[][]mat, int teta)** מקבל את מטריצת עלויות כמערך דו-ממדי של קדקודים ומספר שלם **teta**.

2.2. פונקציה שמחזירה את מספר המסלולים הקצרים ביותר (ע"פ עלות בלבד):

```
public int getNumOfCheapestPaths()
```

2.3. פונקציה שמחזירה את מספר המסלולים האופטימאליים (ע"פ עלות ומספר פניות):

```
public int getNumOfOptimalPaths()
```

2.4. פונקציה שמחזירה עלות מינימאלית:

```
public double getCheapestPrice()
```

2.5. פונקציה שמחזירה את מספר הפניות המינימאלי.

```
public int printNumOfTurns()
```

2.6. פונקציה שמחזירה את כל המסלולים הקצרים ביותר כאשר מספר המסלולים לא עולה על **teta**.

```
public ArrayList<String> getAllCheapestPaths()
```

את המסלולים צריך להציג בצורה הבאה: 010101, כאשר 0 מסמן צעד אחד ימינה ו – 1 מסמן צעד אחד כלפי מעלה.

2.7. פונקציה שמחזירה את כל המסלולים האופטימאליים.

```
public ArrayList<String> getAllOptimalPaths()
```

<u>ניתן להשתמש בדוגמה הנ"ל לבדיקת נכונות התוכנית.</u>

מספר המסלולים הקצרים ביות הוא 4, העלות של המסלול הקצר ביותר היא 20 והמסלולים הם:
מסלול 1: 010101 , עלות המסלול: 1+4+5+1+1+8=20
מסלול 2: 100101 , עלות המסלול: 3+2+5+1+1+8=20
מסלול 3: 010110 , עלות המסלול: 1+4+5+1+4+5=20
מסלול 4: 100110 , עלות המסלול: 3+2+5+1+4+5=20
מספר המסלולים האופטימאליים הוא 1, העלות של המסלול האופטימאלי היא ת 20, מספר פניות 3.
המסלול האופטימאלי הוא מסלול 4: 100110.

חלק ב' מסלולים בעלי עלות משנית

בחלק זה עליכם למצוא את מספר מסלולים בעלי עלות משנית בין נקודה (0,0) לנקודה (M,N) ולבנות את כל המסלולים בעלי עלות משנית, כאשר מספר המסלולים האלה לא עולה על אותו מספר נתון **teta**.
עליכם גם לבחור בין המסלולים בעלי עלות משנית את המסלולים בעלי מספר מינימאלי של פניות.

<u>הנחיות לתכנות:</u>
עליכם להוסיף למחלקה **BestPath** פונקציות הבאות שדומות לפונקציות של חלק א', אך הן מתייחסות **לעלות משנית** במקום עלות קטנה ביותר.

1. פונקציה שמחזירה את מספר המסלולים בעלי עלות משנית.

```
public int getNumOfCheapestPaths2()
```

2. פונקציה שמחזירה את מספר המסלולים בעלי עלות משנית שמספר פניות בהם הוא קטן ביותר.

```
public int getNumOfOptimalPaths2()
```

3. פונקציה שמחזירה את העלות המשנית:

```
public double getCheapestPrice2()
```

4. פונקציה שמחזירה את מספר הפניות המינימאלי במסלולים בעלי עלות משנית.

```
public int getNumOfTurns2()
```

5. פונקציה שמחזירה את כל המסלולים בעלי עלות משנית, כאשר מספר המסלולים לא עולה על **teta**.

```
public ArrayList<String> getAllCheapestPaths2()
```

6. פונקציה שמחזירה את כל המסלולים בעלי עלות משנית שמספר פניות בהם הוא קטן ביותר.

```
public ArrayList<String> getAllOptimalPaths2()
```

<u>ניתן להשתמש בדוגמה הנ"ל לבדיקת נכונות התוכנית.</u>

יש רק מסלול אחד בעל עלות משנית.
המסלול: 110100 עלות המסלול: 3+5+4+1+3+5=21, מספר פניות 3.

עבודה מהנה!