יסודות מערכות פתוחות
פתרון תרגיל מס' 9

.1

כתוב/כתבי תכנית ב- Bash (דהינו קובץ Script) בשם P9.1 שמקבלת קלט שם קובץ ומדפיסה לפלט את תוכן הקובץ כך שכל העמודות של הקובץ מיושרות לימין לפי המילה הארוכה ביותר בכל עמודה. בין המילים הארוכות ביותר בעמודות עוקבות ישנו תו רווח אחד בדיוק.

מומלץ להשתמש בפקודה printf של awk לישור עמודות לימין.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
abcd   123.567  8812  aabbcc
  xz    1.56    12
xyzde  7.1     144
1256
12 12 12
```

לאחר הפעלת התכנית על ידי הפקודה P9.1 F1 יתקבל הפלט:

```
 abcd 123.567 8812  aabbcc
   xz    1.56   12
xyzde     7.1  144
 1256
   12      12   12
```

פתרון שאלה 1

```
rm -f tmp
maxWords=0
while read line
 do
  current=$(echo $line | tr " " "\n" | wc -l)
  if [ $current -gt $maxWords ]
   then
    maxWords=$current
  fi
 done <$1

for x in $(seq $maxWords)
 do
  maxWordLength=0
  while read line
   do
    wordLength=$(echo $line |cut -d" " -f$x| wc -c)
    if [ $wordLength -gt $maxWordLength ]
     then
       maxWordLength=$wordLength
    fi
   done <$1
  maxWordLength=$[$maxWordLength-1]
  echo $maxWordLength >> tmp
 done

while read line
 do
  numOfWords=$(echo $line | wc -w)
  for i in $(seq $numOfWords)
   do
    s=$(echo $(cat tmp) | cut -d" " -f$i)
    echo $line | awk '{printf "%'$s's",$'$i'}'
    if [ $i -lt $numOfWords ];then
     echo -n " "
    else
     echo
    fi
   done
 done<$1
```

.2

הגדרה: נגדיר שמטריצה ריבועית היא טובה, אם סכומי העמודות שלה יוצרים סדרה חשבונית.

לדוגמה, המטריצה הבאה היא טובה:

```
1 1 1.2
2 5 2.8
3 3 8
```

כי סכום העמודה הראשונה הוא: 6
סכום העמודה השניה הוא 9
וסכום העמודה השלישית הוא 12

והמספרים 6,9,12 יוצרים סדרה חשבונית.

כתוב/כתבי תכנית ב- Awk בשם P9.2 שמקבלת קלט שם קובץ שמכיל מטריצה ריבועית של מספרים, ומדפיסה לפלט YES אם הקובץ מכיל מטריצה ריבועית טובה, אחרת התכנית מדפיסה NO.

שימו לב שהפלט הוא שורה אחת בדיוק כך שבסוף המחרוזת שתודפס לפלט (דהינו המחרוזת YES או המחרוזת NO) יש לדאוג לכך שיודפס גם תו קפיצת שורה.

לדוגמה, נניח שתוכן הקובץ F1.1 הוא:

```
1 1 1.2
2 5 2.8
3 3 8
```

לאחר הפעלת התכנית על ידי הפקודה P9.2 F1.1 יתקבל הפלט:

```
YES
```

לדוגמה, נניח שתוכן הקובץ F1.2 הוא:

```
1 1 1.2 6
2 5 2.8 -1
0 3 8 1
3 0 0 9
```

לאחר הפעלת התכנית על ידי הפקודה P9.2 F1.2 יתקבל הפלט:

```
YES
```

נניח שתוכן הקובץ F1.3 הוא:

```
1 1 5
2 5 4
1 2 2
```

לאחר הפעלת התכנית על ידי הפקודה P9.2 F1.3 יתקבל הפלט:

```
NO
```

פתרון שאלה 2

```
#!/bin/awk -f

{ for (j=1; j<=NF; j++) {
  A[FNR,j]=$j
  }
}
END { for (j=1; j<=FNR; j++) {
       for (i=1; i<=FNR; i++) {
            B[j]+=A[i,j]
         }
       }
    d=B[2]-B[1]
    result="YES"
    for (j=3; j<=FNR; j++) {
      if((B[j]-B[j-1]) != d) {
        result="NO"
        break
      }
    }
    print result
}
```

3. שאלה זו הופיע במבחן מועד א 2015 (החומר לפתרון שאלה זו נמצא בחומר ללימוד עצמי כפי שמופיע באתר הקורס).

נגדיר שמילה היא רצף של תווים ללא תווי רווח וסוף שורה.

כתוב תוכנית Script ב- sed בשם P3 שמקבלת כפרמטר שם קובץ (בהמשך נקרא לו קובץ1) ושני מספרים (בהמשך נקרא להם מספר1 ומספר2) ומדפיסה לפלט את השורות בקובץ1 שנמצאות בין שורה מספר 1 לשורה מספר2 (כולל שורה מספר1 ושורה מספר2), כאשר בשורות אלה אם המילה הראשונה בשורה אינה מכילה ספרות כלל אז היא מוחלפת במילה האחרונה בשורה.

על המבנה של התכנית P2 להיות כדלהלן:

התכנית מכילה שורה אחת או יותר של פקודות sed לדוגמה:

```
sed  s/dog/cat/  $1
```

אין להשתמש בפקודות שאינן של sed ושאינן מתחילות ב- sed.

מותר לכתוב לקובץ ביניים כמו למשל:

```
sed  s/dog/cat/  $1 >| tmp
```

מותר גם לקרוא מקובץ ביניים כמו למשל:

```
sed  s/dog/cat/  < tmp
```

אסור להשתמש ב- pileline זאת אומרת המבנה הבא אסור:

```
sed  s/dog/cat  $1 |  sed  s/abc/def
```

לדוגמה, נניח שתוכן הקובץ F3 הוא:

```
abc2 1 def2 3 abc2
dea 123 zy45
cdeabc1d rq12345  cd
    abcabc  456ab
xyz
hyz dd
```

לאחר הפעלת התכנית ע"י הפקודה:

```
P9.3 F3 2 5
```

יתקבל הפלט:

```
zy45 123 dea
cdeabc1d rq12345  cd
    456ab  abcabc
xyz
```

פתרון שאלה 3

```
sed -n $2','$3'p' $1 >| tmp
sed -r 's/^([ ]*)([^ 0-9]+)( )([^ ]+)([ ]*)$/\1\4\3\2\5/' tmp >| tmp1
sed -r 's/^([ ]*)([^ 0-9]+)([ ]+.*[ ]+)([^ ]+)([ ]*)$/\1\4\3\2\5/' tmp1
```

11. הסבר לפתרון נמצא בסוף הסרטון של הרצאה