

יטודות מערכות פתוחות
פתרון תרגיל מס' 9

שימו לב: כל ההערות שבתחילת תרגילים 8-1 תקפות גם לתרגיל זה.

.1

הגדרה: נגדיר את הממוצע של עמודה i במטריצה כסכום המספרים בעמודה i של המטריצה לחלק למספר המספרים בעמודה זו.

לדוגמה עבור המטריצה הבאה:

10 20 30
5 8
9

הממוצע של עמודה 1 הוא 8
הממוצע של עמודה 2 הוא 14
הממוצע של עמודה 3 הוא 30

כתוב/כתבי תכנית ב- Awk בשם P9.1 שמקבלת כפרמטרים שמות קבצים שמכילים מטריצות (לא בהכרח ריבועיות) ומדפיסה לפלט עבור כל קובץ את שם הקובץ ולאחר מכן את סכום המספרים בכל עמודה שגדולים או שווים לממוצע העמודה. על פורמט הפלט להיות כפי שמתואר בדוגמה שבהמשך.

על סדר השורות בפלט להיות לפי סדר הפרמטרים לתכנית.

לדוגמה, נניח שתוכן הקובץ F1.1 הוא:

10 20 30
5 8
9

נניח שתוכן הקובץ F1.2 הוא:

1 1 2 2 3
5
6 6
1 1 1 1 1

נניח שתוכן הקובץ F1.3 הוא:

1 2 3 4 5
6 7 8 4 10
8
10 9 8 7 6

לאחר הפעלת התכנית על ידי הפקודה:

P9.1 F1.1 F1.2 F1.3

יתקבל הפלט:

F1.1: 19 20 30

F1.2: 11 6 2 2 3

F1.3: 18 16 16 7 10

```
#!/bin/awk -f
function get_column(file,i,result){
    result=""
    while (getline<file){
        if (i<=NF) result=result" "$i
    }
    close(file)
    return substr(result,2)
}

function sum_above_average(col,A,n,sum,i,avg){
    n=split(col,A," ")
    sum=0
    for (i=1;i<=n;i++) sum+=A[i]
    avg=sum/n
    sum=0
    for (i=1;i<=n;i++) {
        if (A[i]>=avg) sum+=A[i]
    }
    return sum
}

function num_cols(file,num_c){
    while (getline<file) {
        if ( num_c < NF) num_c=NF
    }
    close(file)
    return num_c
}

function calc_file(file,n,i,col,result){
    n=num_cols(file)
    result=file":"
    for (i=1;i<=n;i++){
        col=get_column(file,i)
        result=result" "sum_above_average(col)
    }
    printf result"\n"
}

BEGIN {
    for (i=1;i<ARGC;i++) calc_file(ARGV[i])
}
```

להלן אותו פתרון בפורמט טקסט.

```
#!/bin/awk -f
function get_column(file,i,result){
    result=""
    while (getline<file){
        if (i<=NF) result=result" "$i
    }
    close(file)
    return substr(result,2)
}
function sum_above_average(col,A,n,sum,i,avg){
    n=split(col,A," ")
    sum=0
    for (i=1;i<=n;i++) sum+=A[i]
    avg=sum/n
    sum=0
    for (i=1;i<=n;i++) {
        if (A[i]>=avg) sum+=A[i]
    }
    return sum
}
function num_cols(file,num_c){
    while (getline<file) {
        if ( num_c < NF) num_c=NF
    }
    close(file)
    return num_c
}
function calc_file(file,n,i,col,result){
    n=num_cols(file)
    result=file":"
    for (i=1;i<=n;i++){
        col=get_column(file,i)
        result=result" "sum_above_average(col)
    }
    printf result"\n"
}
BEGIN {
    for (i=1;i<ARGC;i++) calc_file(ARGV[i])
}
```

.2

כתוב/כתבי תכנית ב- Awk בשם P9.2 שמקבלת כפרמטרים שמות קבצים שמכילים מטריצות (לא בהכרח ריבועיות) ומדפיסה לפלט שורה אחת עבור כל עמודה, שמכילה את הסכומים של כל האיברים שנמצאים בעמודה זו בכל אחד מהקבצים (שבהם עמודה זו קימת) ובסוף השורה YES אם סדרת הסכומים היא סדרה חשבונית או NO אחרת. על פורמט הפלט להיות כפי שמתואר בדוגמה שבהמשך.

סדרה שמכילה מספר אחד בלבד או שני מספרים בלבד נחשבת סדרה חשבונית.

על סדר השורות להיות לפי מספרי העמודות בסדר עולה, ובכל שורה על סדר האיברים להיות לפי הסדר של הפרמטרים לתכנית.

לדוגמה, נניח שתוכן הקובץ F2.1 הוא:

```
4 20 24
4
2 5 6 40 10
```

תוכן הקובץ F2.2 הוא:

```
6 2 13
4 5 7
2 10
8 3 20 50 20 70
```

תוכן הקובץ F2.3 הוא:

```
20 13
4 2
6
```

תוכן הקובץ F2.4 הוא:

```
15 1 20 60
25 9 5
```

לאחר הפעלת התכנית על ידי הפקודה:

```
P9.2 F2.1 F2.2 F2.3 F2.4
```

יתקבל הפלט:

```
1#F2.1:10 F2.2:20 F2.3:30 F2.4:40 YES
2#F2.1:25 F2.2:20 F2.3:15 F2.4:10 YES
3#F2.1:30 F2.2:40 F2.4:25 NO
4#F2.1:40 F2.2:50 F2.4:60 YES
5#F2.1:10 F2.2:20 YES
6#F2.2:70 YES
```

הסבר לפלט:

מבנה השורה הראשונה בפלט:

בתחילת השורה מופיע המספר 1 ולאחריו # שורה זו מתארת סכום המספרים בעמודה 1 בכל אחד מהקבצים. בקובץ F2.1 סכום המספרים בעמודה 1 הוא $4+4+2$ ששורה ל- 10 ולכן מופיע הזוג F2.1:10 בהמשך מופיע הזוג F2.2:20 כי בקובץ F2.2 סכום המספרים בעמודה 1 הוא $2+5+10+3$ ששורה ל- 20 וכן הלאה. בסוף השורה מופיע תו רווח ולאחריו YES כי סדרת הסכומים: 10 20 30 40 היא סידרה חשבונית.

מבנה השורה החמישית בפלט:

השורה נבנית באופן דומה לשורה הראשונה. בשורה זו לא מופיעים הקבצים F2.3 ו-F2.4 כי לקבצים אלה אין עמודה חמישית.

```
#!/bin/awk -f
function get_column(file,i,result){
    result=""
    while (getline<file){
        if (i<=NF) result=result" "$i
    }
    close(file)
    return substr(result,2)
}
function sum_col(file,i,n,col,sum,A){
    col=get_column(file,i)
    n=split(col,A," ")
    sum=0
    for (i=1;i<=n;i++) sum+=A[i]
    return sum
}
function num_cols(file,num_c){
    while (getline<file) {
        if ( num_c < NF) num_c=NF
    }
    close(file)
    return num_c
}
function check_sidra(sidra,n,A,diff,i){
    n=split(sidra,A," ")
    if (n<=2) return "YES"
    diff=A[2]-A[1]
    for (i=3;i<=n;i++){
        if ((A[i]-A[i-1])!=diff) return "NO"
    }
    return "YES"
}
function calc_col(i,result,sidra,x){
    result=i"#";sidra=""
    for (x=1;x<ARGC;x++) {
        if (num_cols(ARGV[x])<i) continue
        result=result" "ARGV[x]":"sum_col(ARGV[x],i)
        sidra=sidra" "sum_col(ARGV[x],i)
    }
    printf result" "check_sidra(sidra)"\n"
}
```

```

BEGIN {
    for (i=1;i<ARGC;i++) {
        if (num_c < num_cols(ARGV[i])) {
            num_c=num_cols(ARGV[i])
        }
    }
    for (i=1;i<=num_c;i++){
        calc_col(i)
    }
}

```

להלן אותו פתרון בפורמט טקסט.

```

#!/bin/awk -f
function get_column(file,i,result){
    result=""
    while (getline<file){
        if (i<=NF) result=result" "$i
    }
    close(file)
    return substr(result,2)
}
function sum_col(file,i,n,col,sum,A){
    col=get_column(file,i)
    n=split(col,A," ")
    sum=0
    for (i=1;i<=n;i++) sum+=A[i]
    return sum
}
function num_cols(file,num_c){
    while (getline<file) {
        if ( num_c < NF) num_c=NF
    }
    close(file)
    return num_c
}
function check_sidra(sidra,n,A,diff,i){
    n=split(sidra,A," ")
    if (n<=2) return "YES"
    diff=A[2]-A[1]
    for (i=3;i<=n;i++){
        if ((A[i]-A[i-1])!=diff) return "NO"
    }
    return "YES"
}
function calc_col(i,result,sidra,x){
    result=i"#";sidra=""
    for (x=1;x<ARGC;x++) {
        if (num_cols(ARGV[x])<i) continue
        result=result" "ARGV[x]":"sum_col(ARGV[x],i)
        sidra=sidra" "sum_col(ARGV[x],i)
    }
    printf result" "check_sidra(sidra)"\n"
}

```



```

BEGIN {
  for (i=1;i<ARGC;i++) {
    if (num_c < num_cols(ARGV[i])) {
      num_c=num_cols(ARGV[i])
    }
  }
  for (i=1;i<=num_c;i++){
    calc_col(i)
  }
}

```

.3

שאלה זו הופיע במבחן מועד ב 2016 (החומר לפתרון שאלה זו נמצא בחומר ללימוד עצמי כפי שמופיע באתר הקורס).

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

כתוב תכנית Script ב- sed בשם P9.3 שמקבלת כפרמטר שם קובץ ומדפיסה לפלט את השורות בקובץ שמקימות את התנאי הבא:
מספר המילים בשורה הוא בדיוק 5 ובנוסף לכך המילה הראשונה זהה למילה האחרונה והמילה השניה זהה למילה הרביעית.

בסוף הפלט מופיע מספר שמציין את מספר השורות בקובץ שמספר המילים שבהן שונה מ- 5.

על המבנה של התכנית P9.3 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 לשרשר 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- pipeline זאת אומרת המבנה הבא אסור:

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

לדוגמה, נניח שתוכן הקובץ F3 הוא:

```
abc2 1 def2 1 abc2
dea 123 123 dea
12321
abc cd abc cd abc
a b c b a
a b c b a 1
a b c d e f
1 2 3 2 2
```

לאחר הפעלת התכנית ע"י הפקודה:

```
P9.3 F3
```

יתקבל הפלט:

```
abc2 1 def2 1 abc2
abc cd abc cd abc
a b c b a
4
```

9.3 פתרון שאלה

```
sed -r '/^[ ]*([ ]+)[ ]+([ ]+)[ ]+[ ]+[ ]+\2[ ]+\1[ ]*$/!d' $1
sed -r '/^[ ]*^[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]*$/d' $1 >| temp3
sed -n '$=' temp3
```

להלן אותו פתרון בפורמט טקסט.

```
sed -r '/^[ ]*([ ]+)[ ]+([ ]+)[ ]+[ ]+[ ]+\2[ ]+\1[ ]*$/!d' $1
sed -r '/^[ ]*^[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]+[ ]*$/d' $1 >| temp3
sed -n '$=' temp3
```