

24.1.2012

פתרון מבחן מועד א'
יסודות מערכות פתוחות
סמסטר חורף, תשע"ב

- משך המבחן: שלוש וחצי שעות.
- יש לענות על כל השאלות.
- מותר השימוש בחומר עזר כלשהו, פרט למחשבים, (מחשבוניס מותר).
- יש להקפיד על כתיבה ברורה ומסודרת של התשובות.

1. (15 נקודות)
חלק א (5 נקודות)

הסעיף הזה לא בחומר של סמסטר חורף תשע"ג (כי לא למדנו expect) לכן לא כתבתי את הפתרון.

חלק ב (10 נקודות)

מה יתקבל בפלט (דהינו יוצג על המסך) לאחר הפעלת קטע הקוד הבא:

```
echo "output 1:"
echo -e "abcd\n123456" >| F1
echo efgh >| F2
echo 78910 >| F3
cut -c2- F1 - F2 < F3

echo "output 2:"
echo "abc@%^def*ABC1      23" | tr -cs a-z0-9 "\n"

echo "output 3:"
echo "abc  shalom" >|F1
echo "  abc  shalom1" >> F1
echo "abc  shalom 123" >> F1
echo "abc  shalom 123 shalom" >> F1
echo "abc  shalom 123 456 shalom" >> F1
egrep "^[ ]*[^ ]+[ ]+shalom($|[ ]+[^ ]+[ ]+shalom)" F1

echo "output 4:"

echo abcdaabcdabcdabcd >| F1
echo 12341112341123411 >> F1
echo aaabbbaaaabba >> F1
egrep "^(.(.+.)\2\1\1\2$" F1
```

```
echo "output 5:"  
echo "1 xyz 6" >|F1  
echo "2 abc 6" >> F1  
echo "3 xyz 2" >> F1  
sort -k 3n,3 -k 2,2 F1 | cut -c1
```

הפלט יהיה:

```
output 1:  
bcd  
23456  
8910  
fgh  
output 2:  
abc  
def  
1  
23  
output 3:  
abc shalom  
abc shalom 123 shalom  
output 4:  
abcdabcdabcdabcd  
12341112341123411  
output 5:  
3  
2  
1
```

אין צורך לרשום איך התקבל הפלט מספיקה תשובה סופית.

2. (10 נקודות)

כתוב תוכנית Script ב- sed בשם P2 שמקבלת כפרמטר שם קובץ (בהמשך נקרא לו קובץ 1). התוכנית מדפיסה לפלט את כל השורות בקובץ שמכילות לפחות 3 ספרות כאשר התוכנית משכפלת בשורות אלה את כל הספרות החל מהספרה הרביעית והלאה.

על המבנה של התוכנית P2 להיות כדלהלן:

שורה ראשונה כמו בכל קובץ script ב- bash מכילה:

```
#!/bin/bash
```

לאחר מכן שורה אחת או יותר של פקודות sed לדוגמה

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

(אין להשתמש בפקודות שאינן של sed).
יותר לכתוב לקובץ ביניים כמו למשל:

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

יותר גם לקרוא מקובץ ביניים כמו למשל:

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

אסור להשתמש ב- pipeline.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
abc 1 def2 3 ab4  
123 zy  
ab1cd2ef3gh4z56  
abc12aa
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P2 F1
```

יתקבל הפלט:

```
abc 1 def2 3 ab44  
123 zy  
ab1cd2ef3gh44z5566
```

השאלה הזאת הופיעה בתרגיל בית מספר 11 כשאלה 4, ולכן לא כתבתי את הפתרון שלה.

ההגדרות הבאות הן עבור שאלות 3 ו-4.

הגדרה: נגדיר את הריבוע החיצוני של מטריצה $n \times n$ כריבוע המורכב מהשורות 1 ו- $n-1$ ומהעמודות 1 ו- n של המטריצה. נגדיר שהריבוע החיצוני של המטריצה הוא טוב אם סכום האיברים שנמצאים בעמודה הראשונה ובשורה הראשונה שווה לסכום של האיברים שנמצאים בעמודה האחרונה ובשורה האחרונה. בחישוב הסכומים הנ"ל האיבר שנמצא גם בשורה הראשונה וגם בעמודה הראשונה מחושב רק פעם אחת, ובאופן דומה האיבר שנמצא גם בעמודה האחרונה וגם בשורה האחרונה מחושב רק פעם אחת.

לדוגמה, עבור המטריצה 4×4 הבאה:

```
5 6 2 3
2 5 6 5
1 1 2 3
1 1 3 4
```

הריבוע החיצוני הוא:

```
5 6 2 3
2      5
1      3
1 1 3 4
```

הריבוע החיצוני הנ"ל הינו טוב. (סכום האיברים בשורה הראשונה ובעמודה הראשונה שווה ל-20 ושווה גם לסכום האיברים בשורה האחרונה ובעמודה האחרונה).

הגדרה: נגדיר שמטריצה B שמימדיה $n-2 \times n-2$ היא צמצום אחד של מטריצה A שמימדיה $n \times n$ אם היא מתקבלת מ- A ע"י הוצאת האיברים בריבוע החיצוני של A . לדוגמה, המטריצה הבאה היא צמצום אחד של המטריצה שלמעלה:

```
5 6
1 2
```

הגדרה: נגדיר שמטריצה A שמימדיה $n \times n$ (עבור n זוגי) היא מטריצה ריבועית טובה אם הריבוע החיצוני שלה הוא טוב, והריבוע החיצוני של המטריצה B שהיא צמצום אחד של A הוא טוב, והריבוע החיצוני של המטריצה C שהיא צמצום אחד של B הוא טוב, וכן הלאה (עד שמגיעים למטריצה 2×2). לדוגמה, המטריצה הבאה היא מטריצת ריבועית טובה:

```
1 0 3 0 1 0
1 2 0 1 0 2
1 2 2 5 1 0
1 0 5 2 4 1
1 1 1 2 1 2
1 0 1 0 3 0
```

3. (20 נקודות)

חלק א (10 נקודות)

כתוב/כתבי תוכנית ב-awk בשם P3.1 שמקבלת פרמטר שם קובץ שמכיל מטריצה ריבועית (כך שמספר האיברים בכל שורה ועמודה הוא זוגי). ניתן להניח שהקובץ קיים ומכיל מטריצה ריבועית של מספרים עם תו רווח אחד או יותר בין המספרים. התוכנית מדפיסה לפלט yes אם הריבוע החיצוני של המטריצה הוא טוב, אחרת התוכנית מדפיסה no.

פתרון חלק א:

```
{
  for (i=1; i<= NF; i++) {
    A[NR,i]=$i
  }
}
END {
  for (i=1; i <=NR; i++) {
    sum1 = sum1 + A[1,i] + A[i,1]
    sum2 = sum2 + A[NR,i] + A[i,NR]
  }
  sum1 = sum1 - A[1,1]
  sum2 = sum2 - A[NR,NR]
  if ( sum1 == sum2 ) { print "yes" }
  else { print "no" }
}
```

חלק ב (10 נקודות)

כתוב/כתבי תוכנית ב-awk בשם P3.2 שמקבלת כפרמטרים רשימת שמות של קבצים שמכילים מטריצות ריבועיות (כך שמספר האיברים בכל שורה ועמודה הוא זוגי). התוכנית מדפיסה לפלט עבור כל קובץ שורה שמכילה את שם הקובץ, לאחריו תו רווח ולאחריו yes אם הריבוע החיצוני של המטריצה שהקובץ מכיל הוא טוב או no אם הריבוע החיצוני של המטריצה שהקובץ מכיל אינו טוב. על סדר השורות בפלט להיות סדר שמות הקבצים ברשימת הפרמטרים.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
1 1 8 9
3 5 6 1
3 1 2 3
1 1 3 4
```

ונניח שתוכן הקובץ A הוא:

```
1 0 3 0 1 0
1 2 0 1 0 2
1 3 2 5 1 0
1 3 5 2 4 1
1 1 1 2 1 2
1 0 1 0 3 0
```

```
awk -f P3.1 F1
```

```
no
```

```
awk -f P3.2 F1 A
```

```
F1 no
```

```
A yes
```

לאחר הפעלת התוכנית P3.1 ע"י הפקודה:

יתקבל הפלט:

לאחר הפעלת התוכנית P3.2 ע"י הפקודה:

יתקבל הפלט:

פתרון חלק ב (בעזרת שימוש בפונקציות)

```
{
for (i=1; i<= NF; i++) {
  A[FILENAME,FNR,i]=$i
  B[FILENAME]=FNR
}
}

END {
  for (x=1; x < length(ARGV); x++) {
    if (check_file(A,ARGV[x],B[ARGV[x]])=="no")
      { print ARGV[x] ": no" }
    else { print ARGV[x] ": yes" }
  }
}

function check_file(A,file,n) {
  sum1=0; sum2=0;
  for (i=1; i <=n; i++) {
    sum1 = sum1 + A[file,1,i] + A[file,i,1]
    sum2 = sum2 + A[file,n,i] + A[file,i,n]
  }
  sum1 = sum1 - A[file,1,1]
  sum2 = sum2 - A[file,n,n]
  if ( sum1 == sum2 ) { return "yes" }
  else { return "no" }
}
```

פתרון חלק ב (ללא שימוש בפונקציות)

```
{
for (i=1; i<= NF; i++) {
  A[FILENAME,FNR,i]=$i
  B[FILENAME]=FNR
}
}

END {
  for (x=1; x < length(ARGV); x++) {
    sum1=0; sum2=0; n=B[ARGV[x]]; file=ARGV[x]
    for (i=1; i <=n; i++) {
      sum1 = sum1 + A[file,1,i] + A[file,i,1]
      sum2 = sum2 + A[file,n,i] + A[file,i,n]
    }
    sum1 = sum1 - A[file,1,1]
    sum2 = sum2 - A[file,n,n]
    if ( sum1 == sum2 ) { print file ": yes" }
    else {print file ": no"}
  }
}
```

4. (30 נקודות)

כתוב/כתבי תוכנית ב- bash בשם P4 שמקבלת כפרמטרים רשימת שמות של קבצים שמכילים מטריצות ריבועיות (כך שמספר האיברים בכל שורה ועמודה הוא זוגי). התוכנית מדפיסה לפלט עבור כל קובץ שורה שמכילה את שם הקובץ, לאחריו תו רווח ולאחריו yes אם המטריצה שהקובץ מכיל היא מטריצה ריבועית טובה או no אם המטריצה שהקובץ מכיל אינה ריבועית טובה. על סדר השורות בפלט להיות לפי שמות הקבצים בסדר לכסיקוגרפי עולה.

לדוגמה, נניח שתוכן הקובץ F2 הוא:

```
1 1 1 1
3 5 6 1
3 1 2 7
1 1 3 4
```

ונניח שתוכן הקובץ B הוא:

```
1 1 1 1
3 5 1 1
3 1 5 0
1 1 3 4
```

ונניח שתוכן הקבצים F1 ו-A הוא כפי שצוין בדוגמאות של שאלה 3.

לאחר הפעלת התוכנית ע"י הפקודה:

P4 F1 F2 A B

יתקבל הפלט:

```
A yes
B yes
F1 no
F2 no
```

פתרון שאלה 4 (בעזרת שימוש בפונקציות)

```
#!/bin/bash
function remove-misgeret {
# the function gets a file name in $1 which represents matrix,
# and removes from it the external misgeret.
# The function returns 0 if the external misgeret is good or
# 1 is the external misgeret is not good

echo -n "" >|tmp
line1=$(head -1 $1)
n=$(echo $line1 | wc -w)
linen=$(head -$n $1 | tail -1)
nminus1=${n-1}
sumline1=${$(echo $line1 | tr " " "+")}
sumlinen=${$(echo $linen | tr " " "+")}
A1n=$(echo $line1 | cut -d" " -f$n)
An1=$(echo $linen | cut -d" " -f1)
sum1=${sumline1+$An1}
sum2=${sumlinen+$A1n}
c=2
while [ $c -lt $n ]
do
linec=$(cat $1 | head -$c | tail -1)
linec2=$(echo $linec | cut -d" " -f2-$nminus1)
echo $linec2 >> tmp
sum1=${sum1 + $(echo $linec | cut -d" " -f1)}
sum2=${sum2 + $(echo $linec | cut -d" " -f$n)}
c=$((c+1))
done
cat tmp >| $1
if [ $sum1 -eq $sum2 ]
then
echo 0
else
echo 1
fi
}
```



```

echo -n "" >|tmp2
for file in "$@"
do
flag=0
n1=$(head -1 $file | wc -w) # we assume that n1 is even and greater
n1=$((n1/2)) # than 1
cat $file >| tmp1
for i in $(seq $n1)
do
flag=$(remove-misgeret tmp1)
if [ $flag -eq 1 ]
then
break
fi
done
if [ $flag -eq 0 ]
then
echo "$file yes" >> tmp2
else
echo "$file no" >> tmp2
fi
done
sort tmp2

```

פתרון שאלה 4 (ללא שימוש בפונקציות)

הפתרון ללא פונקציות הוא אותו פתרון כמו הפתרון שלמעלה, פרט לכך שבמקום להשתמש בפונקציה remove-misgeret משתמשים בקובץ סקריפט נפרד בשם זה (שתוכנו הוא כמו תוכן הפונקציה remove-misgeret).

5. (25 נקודות) נקודות

כתוב/כתבי תוכנית ב-Bash (דהינו קובץ Script) בשם P5 שמקבלת כפרמטרים שם תיקיה (בהמשך נקרא לה תיקיה1) ולאחריה רשימת שמות של קבצים (בהמשך נקרא לה רשימה1).

התוכנית מדפיסה לפלט שורה אחת עבור כל שם קובץ שמופיע ברשימה1 שמכילה את שם הקובץ, לאחריו תו רווח אחד ולאחריו מספר המציין את מספר הפעמים ששם קובץ זה מופיע בתוך תיקייה1 (בעומק כלשהו).

על שורות הפלט להיות ממוינות בסדר מספרי עולה לפי המספר שמופיע בכל שורה. במידה ולשתי שורות יש אותו מספר אז המיון ביניהן יהיה לפי שמות הקבצים בסדר לכסיקוגרפי עולה.

לדוגמא, לצורך הצגת מבנה הקבצים בתיקיה d1 נניח שלאחר ביצוע הפקודה tree d1 התקבל הפלט הבא:

```
d1
|-- a
|-- b
|-- c
|-- d
|-- d1
| |-- b
| |-- d2
| | |-- b
| | |-- c
| | |-- d
| | `-- f1
| |-- d3
| | |-- c
| | `-- f1
| `-- f1
`-- f1
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P5 d1 f1 a b c d
```

יתקבל הפלט:

```
a 1
d 2
b 3
c 3
f1 4
```

פתרון שאלה 5 (בעזרת שימוש ב- sort -k)

```
#!/bin/bash
echo -n "" >|tmp
dir="$1"
shift
for file in "$@"
do
n=$(find "$dir" -type f -name "$file" | wc -l)
echo "$file" $n >> tmp
done
sort -k 2n,2n -k 1,1 tmp
```

פתרון שאלה 5 (ללא שימוש ב- sort -k)

```
#!/bin/bash
echo -n "" >|tmp
dir="$1"
shift
for file in "$@"
do
n=$(find "$dir" -type f -name "$file" | wc -l)
echo $n "$file" >> tmp
done
max=$(sort -n tmp | tail -1 | cut -d" " -f1)
for i in 0 $(seq $max)
do
num=$(egrep -c "^$i " tmp)
if [ $num -ge 1 ]
then
for fname in $(egrep "^$i " tmp | cut -d" " -f2 | sort)
do
echo "$fname" $i
done
fi
done
```

בהצלחה !