השאלות הבאות הופיעו במבחן מועד א 2013

2. (10 נקודות)

כתוב תוכנית Script - ב sed בשם P2 שמקבלת כפרמטרים שם קובץ (בהמשך נקרא לו קובץ1) ומחרוזת (בהמשך נקרא לה מחרוזת1).
התוכנית מדפיסה לפלט את השורות בקובץ1 שמכילות את מחרוזת1 (באיזשהו מקום בשורה) כאשר בשורות אלה מתבצע שיכפול של המילה השנייה, כפי שמודגם בדוגמה שלהלן. (ניתן להניח שבכל שורה יש לפחות שתי מילים).

לדוגמה, נניח שתוכן הקובץ F2 הוא:

```
abc 1 def2 3 ab4
123 zy
abc1d rq rq
abcabc 12
```

לאחר הפעלת התוכנית ע"י הפקודה:

P2 F2 abc

יתקבל הפלט:

```
abc 1 1 def2 3 ab4
abc1d rq rq rq
abcabc 12 12
```

פתרון שאלה 2:

```
sed '/'$2'/!d' $1 >| tmp
sed 's/^\([ ]*\)\([^ ]\+\)\([ ]\+\)\([^ ]\+\)\(.*\)$/\1\2\3\4 \4\5/' < tmp
```


3. (20 נקודות)

חלק א (15 נקודות)

כתוב/כתבי תוכנית ב- awk בשם P3.1 שמקבלת פרמטר שם קובץ ומדפיסה לפלט מספר שמציין את סכום הספרות הגדול מבין כל המספרים שמופיעים בקובץ.

פתרון שאלה 3 חלק א:

```
{
  for (i=1; i<= NF; i++) {
  A[NR,i]=$i
  }
  B[NR]=NF
}
END {
 max=0
 for (i=1; i<= NR; i++) {
   for (j=1; j<= B[i]; j++) {
    s=A[i,j]
    if (gsub("[^0-9]","&",s)==0){
      if (sum_digits(s) > max) {
         max=sum_digits(s)
      }
    }
  }
 }
 print max
}

function sum_digits(s) {
  split(s,C,"")
  sum=0
  for (y in C) {
    sum += C[y]
  }
  return sum
 }
```

<u>חלק ב (5 נקודות)</u>

כתוב/כתבי תוכנית ב- awk בשם P3.2 שמקבלת כפרמטרים רשימת שמות של קבצים ומדפיסה לפלט שורה אחת עבור כל קובץ שברשימת הקבצים שמכילה את שם הקובץ, לאחריו תו רווח ולאחריו מספר שמציין את סכום הספרות הגדול ביותר מבין כל המספרים שמופיעים בקובץ.
על סדר הקבצים בפלט להיות לפי סדר הופעתם ברשימת הפרמטרים.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
1234 99999a 77b22
abcabc   888 99
66666
```

ונניח שתוכן הקובץ A הוא:

```
12yy34 99999a 17522
abcabc   88a8 99
66c666  111
2222222
```

לאחר הפעלת התוכנית P3.1 ע"י הפקודה:

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

יתקבל הפלט:

30

לאחר הפעלת התוכנית P3.2 ע"י הפקודה:

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

יתקבל הפלט:

F1 30
A 18

פתרון שאלה 3 חלק ב:

```
{
 for (i=1; i<= NF; i++) {
  A[FILENAME,FNR,i]=$i
 }
 B[FILENAME,FNR]=NF
 D[FILENAME]=FNR
}
END {
 for (x=1; x < length(ARGV); x++) {
  print ARGV[x],calc_file(ARGV[x]);
 }
}

function calc_file(file) {
 max=0
 for (i=1; i<= D[file]; i++) {
  for (j=1; j<= B[file,i]; j++) {
   s=A[file,i,j]
   if (gsub("[^0-9]","&",s)==0){
    if (sum_digits(s) > max) {
     max=sum_digits(s)
    }
   }
  }
}
 return max
}

function sum_digits(s) {
 split(s,C,"")
 sum=0
 for (y in C) {
   sum += C[y]
 }
 return sum
}
```

5. (25 נקודות)

כתוב/כתבי תוכנית ב- Bash (דהינו קובץ Script) בשם P5 שמקבלת כפרמטרים רשימת תיקיות. התוכנית מדפיסה לפלט את שמות כל הקבצים שמופיעים לפחות פעם אחת (בעומק כלשהו) בכל אחת מהתיקיות שברשימה. כל שם קובץ יופיע בפלט בשורה נפרדת. אין חשיבות לסדר בין שמות הקבצים בפלט.

לדוגמה, להלן מבנה תיקיות d1 d2 d3 כפי שמתקבל על ידי הפעלת הפקודה tree על תיקיות אלו. בתאור המבנה של התיקיות יש להניח ששם שאין מתחתיו קבצים מתאר קובץ (ולא תיקיה).

```
d1
|-- A
|-- AA
|   |-- F1
|   `-- F2
|-- CC
|-- DDD
`-- EE
    `-- F1

d2
|-- A1
|   |-- F1
|   `-- F2
|-- CC
`-- XXX

d3
|-- B
|   |-- F1
|   `-- F2
|-- CC
|-- F1
`-- XXX
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P5 d1 d2 d3
```

יתקבל הפלט:

```
F1
F2
CC
```


פתרון שאלה 5:

```
echo -n "" >| tmp
num_dirs=$#
for dir in "$@"
 do
  find $dir -type f -exec get_file_name {} \; \
  | sort -u >> tmp
 done
sort tmp | uniq -c >| tmp1
while read line
 do
  n=$(echo $line | cut -d" " -f1)
  if [ $n -eq $num_dirs ]
    then
      echo $line | cut -d" " -f2
  fi
 done<tmp1
```

כאשר תוכן הקובץ get_file_name הוא:

```
echo $1 | tr "/" "\n" | tail -1
```

2. (10 נקודות)

כתוב תוכנית Script - ב sed בשם P2 שמקבלת כפרמטרים שם קובץ (בהמשך נקרא לו קובץ1). התוכנית מדפיסה לפלט את השורות בקובץ1 שמכילות שתי מילים זהות רצופות (כאשר ישנו רווח אחד או יותר בין המילים).

לדוגמה, נניח שתוכן הקובץ F2 הוא:

```
abc 1 def2  abc 3 ab4
123   123   zy
abc1d rq rq rq 12
abcabc 123123
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P2 F2
```

יתקבל הפלט:

```
123   123   zy
abc1d rq rq rq 12
```

פתרון שאלה 2:

```
sed '/\(^\|[ ]\+\)\([^ ]\+\)\([ ]\+\)\2\([ ]\+\|$\)/!d' $1
```

אותו הפתרון בפונט יותר גדול:

```
sed '/\(^\|[ ]\+\)\([^ ]\+\)\([ ]\+\)\2\([
]\+\|$\)/!d' $1
```


3. (20 נקודות)

הגדרה: יהי F קובץ כלשהו שמכיל מספרים. נאמר שמספר שמופיע בקובץ F בשורה i ועמודה j הוא **טוב** אם הוא גדול יותר מהמספר שמופיע בקובץ F בשורה 1 בעמודה j. במקרה ששורה 1 מכילה פחות מ- j מספרים, דהינו במקרה שבשורה 1 עמודה j לא קיימת, אזי כל מספר שמופיע בעמודה j בקובץ F באיזושהי שורה הוא טוב.
לדוגמה המספרים הטובים בקובץ F1 שמתואר בהמשך הם: 7,8,3,13

חלק א (15 נקודות)

כתוב/כתבי תוכנית ב- awk בשם P3.1 שמקבלת פרמטר שם קובץ שמכיל מספרים עם רווח אחד או יותר בין המספרים בכל שורה. התוכנית מדפיסה לפלט את סכום המספרים הטובים בקובץ.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
12 6  4  100
1 7 8 50 3
13 5
2
```

לאחר הפעלת התוכנית P3.1 ע"י הפקודה:

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

יתקבל הפלט:

```
31
```

הפלט הנ"ל התקבל כתוצאה מחישוב הסכום הבא: 7+8+3+13

פתרון שאלה 3 חלק א:

```
{
  for (i=1; i<= NF; i++) {
   A[NR,i]=$i
  }
  B[NR]=NF
}
END {
 sum=0
 for (i=2; i<= NR; i++) {
   for (j=1; j<= B[i]; j++) {
    if (j > B[1] || A[i,j] > A[1,j]){
        sum += A[i,j]
     }
    }
  }
 print sum
}
```

חלק ב (5 נקודות)

כתוב/כתבי תוכנית ב- awk בשם P3.2 שמקבלת כפרמטרים רשימת שמות של קבצים ומדפיסה לפלט שורה אחת עבור כל קובץ שברשימת הקבצים שמכילה את שם הקובץ, לאחריו תו רווח ולאחריו מספר שמציין את סכום המספרים הטובים בקובץ. על סדר הקבצים בפלט להיות לפי סדר הופעתם ברשימת הפרמטרים.

לדוגמה, נניח שתוכן הקובץ F1 הוא כפי שתואר לעיל
ונניח שתוכן הקובץ A הוא:

```
10 1 200
1 7 3 30
12
```

לאחר הפעלת התוכנית P3.2 ע"י הפקודה:

```
awk –f P3.2 F1 A
```

יתקבל הפלט:

```
F1 31
A 49
```

פתרון שאלה 3 חלק ב:

```
{
 for (i=1; i<= NF; i++) {
  A[FILENAME,FNR,i]=$i
 }
 B[FILENAME,FNR]=NF
 D[FILENAME]=FNR
}
END {
 for (x=1; x < length(ARGV); x++) {
  print ARGV[x],calc_file(ARGV[x]);
 }
}

function calc_file(file) {
 sum=0
 for (i=2; i<= D[file]; i++) {
   for (j=1; j<= B[file,i]; j++) {
    if (j > B[file,1] || A[file,i,j] > A[file,1,j]){
         sum += A[file,i,j]
       }
     }
  }
 return sum
}
```


4. (30 נקודות)

הגדרה: נאמר שמטריצה ריבועית M היא **מאוזנת** אם סכום המספרים בשורה ה- i במטריצה M שווה לסכום המספרים בעמודה ה- i במטריצה. נגדיר את ה- **סכום שורה-עמודה מכסימאלי** של המטריצה M כסכום הגדול ביותר של המספרים בשורה או עמודה במטריצה.

לדוגמה, הקובץ E שמתואר בדוגמה בהמשך מכיל מטריצה מאוזנת, וסכום השורה-עמודה מכסימלי שלו הוא 270.

כתוב/כתבי תוכנית ב- bash בשם P4 שמקבלת כפרמטרים רשימת שמות של קבצים שמכילים מטריצות ריבועיות של מספרים (עם רווח אחד או יותר בין המספרים בשורה). התוכנית מדפיסה לפלט שורה עבור כל קובץ שמכילה את שם הקובץ, לאחריו תו רווח אחד, לאחריו את הסכום שורה-עמודה מקסימאלי של המטריצה שהקובץ מכיל, לאחריו תו רווח אחד, ולבסוף YES אם הקובץ מכיל מטריצה מאוזנת או NO אם הקובץ אינו מכיל מטריצה מאוזנת.
בנוסף בשורת הפלט האחרונה התוכנית מדפיסה את שמות הקבצים שהסכום שורה-עמודה מקסימאלי שלהם הוא הגדול ביותר. שמות הקבצים שמודפסים בשורת הפלט האחרונה ממוינים לפי סדר לכסיקוגרפי עולה.

לדוגמה, נניח שנתונים הקבצים הבאים שתוכנם הוא:

```
  A       B          C       K       E
10 20    2 3 4     20 200    270    1 2 3 4
20 40    5 4 6     200 70           2 1 4 4
         40 2 7                     3 2 1 2
                                    4 6 0 260
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P4 A B C K E
```

יתקבל הפלט:

```
A 60 YES
B 49 NO
C 270 YES
K 270 YES
E 270 YES
C E K
```

<u>הערה</u>: בפתרון המתואר להלן של שאלה 4, יש שימוש בפונקציות ב- bash.
מהפתרון הזה ניתן לקבל פתרון של השאלה ללא שימוש בפונקציות על ידי שימוש בקובץ script נפרד עבור כל אחת מהפונקציות שמופיעות בפתרון המתואר להלן.

פתרון שאלה 4 :

```
function calc_sum {
 sum=0
 for num in $@
  do
   sum=$[$sum+$num]
  done
  echo $sum
}

function calc_row {
 file=$1
 row=$2
 R=$(head -$row $file | tail -1)
 echo $(calc_sum $R)
}


function calc_col {
 file=$1
 col=$2
 C=""
 while read line
 do
  C="$C $(echo $line | cut -d" " -f$col)"
done<$file
 echo $(calc_sum $C)
}
```



```
function calc {
 file=$1
 n=$(head -1 $file | wc -w)
 max=$(calc_row $file 1)
 for i in $(seq $n)
  do
   if [ $(calc_row $file $i) -gt $max ]
     then
       max=$(calc_row $file $i)
   fi
   if [ $(calc_col $file $i) -gt $max ]
     then
       max=$(calc_col $file $i)
   fi
  done
  echo $max
}

function check {
 file=$1
 n=$(head -1 $file | wc -w)
 Flag=0
 for i in $(seq $n)
  do
   if [ $(calc_row $file $i) -ne $(calc_col $file $i) ]
     then
       Flag=1
   fi
  done
 if [ $Flag -eq 0 ]
  then
    echo YES
  else
    echo NO
 fi
}
```

```
max1=$(calc $1)
max_file_list=$1
for file in "$@"
  do
   echo $file $(calc $file) $(check $file)
   if [ $(calc $file) -gt $max1 ]
    then
     max1=$(calc $file)
     max_file_list=$file
     continue
   fi
   if [ $(calc $file) -eq $max1 ]
    then
     max_file_list="$max_file_list $file"
   fi
  done
list=$(echo $max_file_list | tr " " "\n" | sort)
echo $list
```


5. (25 נקודות)

כתוב/כתבי תוכנית ב- Bash (דהינו קובץ Script) בשם P5 שמקבלת כפרמטרים רשימת קבצים לאחריה המילה dirs ולאחריה רשימת תיקיות.
התוכנית מדפיסה לפלט שורה אחת עבור כל קובץ שמכילה את שם הקובץ, לאחריו תו רווח אחד, לאחריו עבור כל אחת מהתיקיות ברשימת התיקיות: שם תיקיה, תו רווח אחד ומספר שמציין את מספר ההופעות של הקובץ בתיקיה.
על סדר השורות בפלט להיות לפי סדר הקבצים ברשימת הפרמטרים, ובכל שורה על סדר התיקיות להיות לפי רשימת הפרמטרים.
לדוגמה, להלן מבנה תיקיות d1 d2 d3 כפי שמתקבל על ידי הפעלת הפקודה tree על תיקיות אלו. בתאור המבנה של התיקיות יש להניח ששם שאין מתחתיו קבצים מתאר קובץ (ולא תיקיה).

```
d1
|-- A
|-- AA
|   |-- F1
|   `-- F2
|-- CC
|-- DDD
`-- EE
    `-- F1
d2
|-- A1
|   |-- F1
|   `-- F2
|-- CC
`-- XXX
d3
|-- B
|   |-- F1
|   `-- F2
|-- CC
|-- F1
`-- XXX
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P5 F1 F2 A XXX dirs d1 d2 d3
```

יתקבל הפלט:

```
F1 d1 2 d2 1 d3 2
F2 d1 1 d2 1 d3 1
A d1 1 d2 0 d3 0
XXX d1 0 d2 1 d3 1
```

פתרון שאלה 5 :

```
function calc {
 file=$1
 dir=$2
 find $dir -type f -name $file | wc -l
}

echo -n "" >| files
echo -n "" >| dirs

for x in $*
 do
  if [ $x = dirs ]
   then
    shift
    break
  fi
  echo $x >> files
  shift
 done

for x in $*
 do
  echo $x >> dirs
done

for file in $(cat files)
 do
  res=$file
  for dir in $(cat dirs)
   do
    res="$res $dir $(calc $file $dir)"
   done
  echo $res
 done
```