יסודות מערכות פתוחות
פתרון תרגיל מס' 7

שימו לב: כל ההערות שבתחילת תרגילים 1-6 תקפות גם לתרגיל זה.

הערה 1: החל מתרגיל זה והלאה, בכל פעם שכתוב **מילה** הכוונה לרצף כלשהו של תווים ללא תווי רווח או סוף שורה. לדוגמה המחרוזת **abc@#&cd** נחשבת למילה אחת.

1. כתוב/כתבי תוכנית ב- **Bash** (דהינו קובץ **Script**) בשם **P7.1** שמקבלת כפרמטרים שימת מילים (בהמשך נקרא לה רשימה 1) לאחריה המילה **dirs**- ולאחריה רשימת תיקיות (בהמשך נקרא לה רשימה 2).
התוכנית מדפיסה לפלט עבור כל מילה שברשימה 1 שורה אחת שמכילה הפרטים הבאים:
המילה עצמה,
תו רווח אחד,
רשימת מספרים (עם תו רווח אחד בדיוק בין המספרים) שמכילה את מספר הקבצים שבהם מופיעה המילה כמילה בכל אחת מהתיקיות שברשימה 2 (לפי סדר התיקיות שברשימה 2).
תו רווח אחד,
המילה **ALL** או המילה **NOTALL**, כאשר המילה **ALL** תודפס אם בכל אחת מהתיקיות שברשימה 2 (בעומק כלשהו) ישנו קובץ שהמילה מופיעה בו כמילה, אחרת תודפס המילה **NOTALL**.

על סדר השורות בפלט להיות לפי סדר המילים שברשימה 1.

ראה/י דוגמה בעמוד הבא.

לדוגמה: נניח שהתיקיות

~basicsys/win18/ex7/d4
~basicsys/win18/ex7/d5
~basicsys/win18/ex7/d6

מכילות את הקבצים הבאים:

~basicsys/win18/ex7/d4/F1 שתוכנו הוא:
abc def gi hi hi
wxyz hi

~basicsys/win18/ex7/d4/c/D1 שתוכנו הוא:
abc hello hi ab ab ab ab 1
ab   12

~basicsys/win18/ex7/d5/d6/H1 שתוכנו הוא:
shalom def ab hi

~basicsys/win18/ex7/d5/G/H1 שתוכנו הוא:
shalom1 def ab1 hello abc
yab
hi def

~basicsys/win18/ex7/d6/A1 שתוכנו הוא:
rami ronit abc
hi

~basicsys/win18/ex7/d6/B/F1 שתוכנו הוא:
@Hello abc
bye
see you later

לאחר הפעלת התוכנית ע"י הפקודה:
P7.1 hello rr hi ab abc -dirs ~basicsys/win18/ex7/d4
~basicsys/win18/ex7/d5 ~basicsys/win18/ex7/d6
יתקבל הפלט:

hello 1 1 0 NOTALL
rr 0 0 0 NOTALL
hi 2 2 1 ALL
ab 1 1 0 NOTALL
abc 2 1 2 ALL

פתרון שאלה 7.1

```
function num_of_files(){
   local word=$1 dir=$2 file n count=0
   for file in $(find $dir -type f); do
      n=$(echo $(cat $file)|tr " " "\n"|egrep -c "^$word$")
      if [ $n -gt 0 ]; then
        ((count++))
      fi
   done
   echo $count
}
words=""
for par in $(echo $@); do
    if [ $1 = "-dirs" ]; then
       shift
       dirs=$(echo "$@")
       break
    fi
    words="$words $1"
    shift
done
for word in $words; do
  result="$word ";flag=true
  for dir in $dirs; do
     num=$(num_of_files $word $dir)
     if [ $num -eq 0 ];then
        flag=false
     fi
     result="$result $num"
  done
  if [ $flag = true ]; then
     result="$result ALL"
  else
     result="$result NOTALL"
  fi
  echo $result
done
```

פתרון שאלה 7.1 בפורמט טקסט:

```
function num_of_files(){
   local word=$1 dir=$2 file n count=0
   for file in $(find $dir -type f); do
      n=$(echo $(cat $file)|tr " " "\n"|egrep -c "^$word$")
       if [ $n -gt 0 ]; then
         ((count++))
       fi
   done
   echo $count
}
words=""
for par in $(echo $@); do
     if [ $1 = "-dirs" ]; then
        shift
        dirs=$(echo "$@")
        break
     fi
     words="$words $1"
    shift
done
for word in $words; do
  result="$word ";flag=true
  for dir in $dirs; do
     num=$(num_of_files $word $dir)
      if [ $num -eq 0 ];then
         flag=false
      fi
     result="$result $num"
  done
  if [ $flag = true ]; then
      result="$result ALL"
  else
     result="$result NOTALL"
  fi
  echo $result
done
```

.2

שאלה זו היא מהחומר ללימוד עצמי שמופיע באתר הקורס.

על התכנית של שאלה זו להיראות כך:
התוכנית 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 להתשמש ב- pileline זאת אומרת המבנה הבא אסור:
```
sed  s/dog/cat  $1 |  sed  s/abc/def
```

כתוב תוכנית Script ב- sed בשם P7.2 שמקבלת כפרמטר שם קובץ (בהמשך נקרא לו קובץ 1) .
התוכנית מדפיסה לפלט את השורות בקובץ1 שמכילות בדיוק 4 מילים, כאשר בכל שורה בפלט מופיעה המילה הראשונה 3 פעמים ברצף (עם תו רווח אחד בין המילים), כפי שמודגם בדוגמה שלהלן.

לדוגמה, נניח שתוכן הקובץ F1 הוא:

```
Hello  hi  123
if a equals b
you
one abc two three four
dany uri four 123
```

לאחר הפעלת התוכנית ע"י הפקודה:
```
P7.2 F1
```
יתקבל הפלט:

```
if if if a equals b
dany dany dany uri four 123
```

פתרון שאלה 7.2

```
sed -r '/^[ ]*([^ ]+[ ]+){3}[^ ]+[ ]*$/!d' $1 >| tmp
sed -r 's/^([ ]*)([^ ]+)([ ]|$)/\1\2 \2 \2\3/' tmp
```

ובפורמט טקסט:

```
sed -r '/^[ ]*([^ ]+[ ]+){3}[^ ]+[ ]*$/!d' $1 >| tmp
sed -r 's/^([ ]*)([^ ]+)([ ]|$)/\1\2 \2 \2\3/' tmp
```

3. כתוב/כתבי תוכנית ב- Bash (דהינו קובץ Script) בשם P7.3 שמקבלת כפרמטרים שם קובץ (בהמשך נקרא לו קובץ 1) ורשימת מספרים (בהמשך נקרא לה רשימה 1) ומדפיסה לפלט את העמודות מתוך קובץ 1 לאחר ישורן לימין או לשמאל לפי המספרים ברשימה 1, כפי שמתואר בדוגמאות שבהמשך. (ניתן להשתמש בפקודה printf של awk לפתרון השאלה).
ניתן להניח שהמספרים ברשימה 1 גדולים יותר מאורכי כל המילים שנמצאות בעמודות המתאימות להם בקובץ 1. (לדוגמה, אם בעמודה הראשונה בקובץ 1 המילה הארוכה ביותר הינה באורך 5, אזי בכל הפעלה של התוכנית המספר הראשון ברשימה 1 יהיה גדול מ- 5).
בנוסף ניתן להניח שבכל שורה בקובץ 1 מספר המילים הוא לפחות כמו מספר המילים ברשימה 1. (לדוגמה אם התוכנית מופעלת עם 4 מספרים ברשימה 1, אזי בקובץ 1 יש לפחות 4 מילים בכל שורה).

דוגמאות,

נניח ש- תוכן קובץ F1 הוא:

```
A   abcd    ddd    eee  zz tt
ab gggwe 12    88  iii jjj
yaara yyzz 12abcd xyz x y z
```

לאחר הפעלת התוכנית ע"י הפקודה:

```
P7.3  F1 -8 -7 6 4
```

יתקבל הפלט:

```
A       abcd       ddd eee
ab      gggwe       12  88
yaara   yyzz   12abcd xyz
```

בדוגמה הנ"ל בין A ל- abcd ישנם 7 רווחים, בין abcd ל- ddd ישנם 6 רווחים ובין ddd ל- eee ישנו רווח אחד.

לאחר הפעלת התוכנית ע"י הפקודה:

```
P7.3  F1 -8 -7 6 4 5
```

יתקבל הפלט:

```
A       abcd      ddd eee   zz
ab      gggwe      12  88  iii
yaara   yyzz   12abcd xyz    x
```

בדוגמה הנ"ל בין A ל- abcd ישנם 7 רווחים, בין abcd ל- ddd
ישנם 6 רווחים, בין ddd ל- eee
ישנו רווח אחד, בין eee ל- zz ישנם 3 רווחים, בין 88 ל- iii
ישנם שני רווחים, בין xyz ל- x ישנם 4 רווחים.

פתרון שאלה 7.3

```
#!/bin/bash
file=$1
shift
x=\"
for z in $@
  do
    x=$x"%"$z"s"
  done
for i in $(seq $#)
  do
    y=$y,\$$i
  done
cat $file |awk '{printf '$x'\n"'$y'}'
```

ובפורמט טקסט:

```
#!/bin/bash
file=$1
shift
x=\"
for z in $@
  do
    x=$x"%"$z"s"
  done
for i in $(seq $#)
  do
    y=$y,\$$i
  done
cat $file |awk '{printf '$x'\n"'$y'}'
```

4. שאלה זו הופיעה במבחן מועד א 2017

נגדיר שמילה נמצאת בקובץ בתחום המוגדר על ידי 4 מספרים i_1 i_2 j_1 j_2 אם המילה נמצאת בקובץ בין שורה i_1 לבין שורה i_2 (כולל שורות i_1 ו- i_2) ובין עמודות j_1 ו- j_2 (כולל עמודות j_1 ו- j_2).

לדוגמה עבור הקובץ הבא:

```
ab    cd    ef  gh
12  xy    zw
67 89 zzz ddd
```

המילה 89 נמצאת בתחום המוגדר על ידי 4 המספרים 2 1 3 2.
המילה zzz אינה נמצאת בתחום המוגדר על ידי 4 המספרים 2 1 3 2.

כתוב/כתבי תכנית ב- bash בשם P7.4 (אין להשתמש בפקודות awk ו- sed בשאלה זו) שמקבלת כפרמטרים 4 מספרים i_1 i_2 j_1 j_2 ולאחריהם רשימת שמות קבצים ומדפיסה לפלט את כל המילים שנמצאות בתחום שמוגדר על ידי 4 המספרים i_1 i_2 j_1 j_2 בכל אחד מהקבצים שברשימת הקבצים. כל מילה תודפס בשורה נפרדת שמכילה את המילה תו רווח אחד ולאחר מכן את מספר ההופעות הכולל של המילה בקבצים שברשימת הקבצים (לא בהכרח בתחום המוגדר על ידי המספרים).

על שורות הפלט להיות ממוינות לפי מספר ההופעות הכולל של המילים בסדר מספרי יורד. (עבור מילים עם אותו מספר הופעות הסדר יהיה לפי סדר לכסיקוגרפי יורד).

ראה/י דוגמה בעמוד הבא.

לדוגמה, נניח שנתונים הקבצים הבאים:

קובץ H שתוכנו הוא:

```
ab gh   cd    gh
12  xy    ef  ef
66 89 zzz ddd
3  44  55   66  ddd
1 66 3 4 66  ef
7 8 9 88 99 gh
```

קובץ G שתוכנו הוא:

```
ab    cd    ef  gh 66
12  xy    zw
67 uu zzz ef
3  99  55   66  88 ddd  99
```

קובץ I שתוכנו הוא:

```
ab    cd    ef  gh
12  xy    zw 88
67 89 zzz  ddd 88
3  44  55   66  77
a b c d   ef
```

לאחר הפעלת התכנית על ידי הפקודה:

```
P7.4 3 6 4 6 H G I
```

מתקבל הפלט:

```
ef 7
66 7
ddd 4
88 4
```

פתרון שאלה 7.4

```
function word_in_files {
  local word=$1 files words
  files=$(echo $@| cut -d" " -f2-)
  echo $(cat $files)|tr " " "\n" | egrep -c "^${word}$"
}
function lines_in_range_i1_i2 {
  local file=$1 i1=$2 i2=$3
  n=$(wc -l < $file)
  if [ $i1 -gt $n ]; then
    return
  fi
  if [ $i2 -gt $n ]; then
     i2=$n
  fi
  head -$i2 $file | tail -$[(i2-i1)+1]
}
function colums_in_range {
 local file=$1 j1=$2 j2=$3 z
 while read line; do
   a=($line)
   z="${a[@]:$[j1-1]:$[(j2-j1)+1]}"
   if [ "$z" != "" ] ; then
     echo ${a[@]:$[j1-1]:$[(j2-j1)+1]}
   fi
 done<$file
}
function words_in_range {
 local file=$1 i1=$2 i2=$3 j1=$4 j2=$5
 lines_in_range_i1_i2 $file $i1 $i2 >| tmp
 echo $(colums_in_range tmp $j1 $j2)
}

i1=$1;i2=$2;j1=$3;j2=$4
files=$(echo $@|cut -d" " -f5-)
for file in $files;do
  words_in_range $file $i1 $i2 $j1 $j2| tr " " "\n"|sort -u
done >|tmp1
num_files=$(echo $files | wc -w)
sort tmp1 | uniq -c | egrep "^[ ]*${num_files}[ ]" >| tmp2
while read line; do
   echo $line
done <tmp2 >| tmp3
words=$(cat tmp3 | cut -d" " -f2)
for word in $words; do
   echo $word $(word_in_files $word $files)
done >|tmp4
sort tmp4 -k 2nr,2 -k 1r,1
```

ובפורמט טקסט:

```
function word_in_files {
  local word=$1 files words
  files=$(echo $@| cut -d" " -f2-)
  echo $(cat $files)|tr " " "\n" | egrep -c "^${word}$"
}
function lines_in_range_i1_i2 {
  local file=$1 i1=$2 i2=$3
  n=$(wc -l < $file)
  if [ $i1 -gt $n ]; then
    return
  fi
  if [ $i2 -gt $n ]; then
     i2=$n
  fi
  head -$i2 $file | tail -$[(i2-i1)+1]
}
function colums_in_range {
 local file=$1 j1=$2 j2=$3 z
 while read line; do
   a=($line)
   z="${a[@]:$[j1-1]:$[(j2-j1)+1]}"
   if [ "$z" != "" ] ; then
     echo ${a[@]:$[j1-1]:$[(j2-j1)+1]}
   fi
 done<$file
}
function words_in_range {
  local file=$1 i1=$2 i2=$3 j1=$4 j2=$5
  lines_in_range_i1_i2 $file $i1 $i2 >| tmp
  echo $(colums_in_range tmp $j1 $j2)
}
i1=$1;i2=$2;j1=$3;j2=$4
files=$(echo $@|cut -d" " -f5-)
for file in $files;do
  words_in_range $file $i1 $i2 $j1 $j2| tr " " "\n"|sort -u
done >|tmp1
num_files=$(echo $files | wc -w)
sort tmp1 | uniq -c | egrep "^[ ]*${num_files}[ ]" >| tmp2
while read line; do
   echo $line
done <tmp2 >| tmp3
words=$(cat tmp3 | cut -d" " -f2)
for word in $words; do
   echo $word $(word_in_files $word $files)
done >|tmp4
sort tmp4 -k 2nr,2 -k 1r,1
```