Задача 5.

```
1       var=1
2       for i in 4 3 2 1
3       do for j
4               do if test $i -gt $#
5                       then var=` expr $var \* $i`
6                               echo $var  $j  >> ff
7                       else continue
8                       fi
9               done
10      done
11
12      while true
13      do echo $*
14              break
15      done
16
17      read k1 k2
18
19      while   cat  ff  |  grep $k2
20      do set $k1 $var
21              shift
22              echo $2
23              grep  $i  ff
24              exit
25              echo $1
26      done
27      wc -l < ff
28      echo END
```

- На ред 1 създаваме променлива var и я инициализираме със стойност 1.
- На ред 2 променливата i експлицитно цикли по подадените на for цикъла аргументи: 4, 3, 2 и 1.
- На ред 3 променливата j имплицитно цикли по аргументите, които са подадени на скрипта при стартирането му (12, 34 и 56). Командата е еквивалентна на for j in "$@".
- На ред 4 се проверява дали 4 е строго по-голямо (gt = greater than) от броя на аргументите на скрипта, който е 3.
- Влизаме в if блок-а (на ред 5) от if-else конструкцията и променливата var се актуализира на 1* 4 = 4.
- На ред 6 записваме стойността на променливата var и стойността на променливата i във файл с име ff. Тоест ще се запише "4 12" във файла ff.
- Ще се върнем още два пъти на ред 3, докато изциклим през всички аргументи на скрипта във вложения цикъл. На всяка итерация на вътрешния цикъл ще актуализираме променливата var като я умножаваме по 4, а променливата j ще обхожда входните аргументи на скрипта. След края на вложения цикъл във файла ff ще имаме записани следните данни:

  4 12
  16 34
  64 56

  При втората итерация на външния цикъл, променливата i вече ще има стойност 3, която няма да е строго по-голяма от броя на аргументите, който все още е 3. При проверката винаги ще влизаме в else блок-а от if-else конструкцията, който само ще продължава цикъла със следващата итерация напред. Аналогичен е сценарият и с всички останали стойности, които ще приема променливата i от външния цикъл.

- На ред 12 имаме безкраен while цикъл, който на ред 13 ще принтира на конзолата (стандартния изход) всички аргументи на скрипта като стринг (между $* и $@ няма разлика, ако се използват без кавички).

- На ред 17 прочитаме от стандартния вход променливите k1 и k2 и им задаваме съответно стойностите 5 и 6.
- На ред 19 в проверката на while цикъла се подава съдържанието на файла ff на командата grep, която търси за съвпадения на всеки ред с подадения ѝ аргумент – стойността на променливата k2, която е 6. Такива съвпадения има на втори и трети ред и тези редове се принтират на конзолата. От друга страна, при намиране на поне едно съвпадение, командата grep връща стойност 0, която контраинтуитивно се интерпретира като true. Следователно ще се влезе в тялото на while цикъла.
- На ред 20 променяме входните параметри на скрипта да са два и да имат стойностите съответно на променливите k1 и var. Следователно сега скрипта ще има два аргумента $1=5 и $2=64.
- На ред 21 изместваме аргументите, по подразбиране с 1 позиция. Командата shift освен, че прави изместването на входните аргументи се грижи и за актуализиране на броя им. Сега вече имаме само един входен аргумент $1=64.
- На ред 22 командата echo $2 ще принтира втория аргумент на скрипта, но то няма такъв и следователно ще се изпечати празен ред.
- На ред 23 търсим за съвпадения на всеки ред от файла ff с подадената като аргумент на командата grep стойност 1 ($i=1). Такива съвпадения има на първи и втори ред и те се принтират на стандартния изход.
- На ред 24 командата exit терминира изпълнението на скрипта и по-долните редове няма да се изпълнят никога.

Отговор:

```
12 34 56
16 34
64 56

4 12
16 34
```

□