Study year: / Študijsko leto: **2022/2023**

Date: / Datum: **17 Feb. 2023**

Exam / Izpit

Computer practicum 1

Računalniški praktikum 1

Instructions:

- The exam contains four assignments that are worth a total of 100 points.
- You have 90 minutes to solve the tasks.
- The use of literature, the Internet and other media are NOT permitted.
- The use of notes on two A4 sheets (written by hand) is allowed.
- Prepare the solution of each task in a (separate) file.
- Solutions (2 bash and 2 C scripts) will be submitted via the e-classroom after the writing time according to the instructions you will receive from the TA.

Navodila:

- Izpit vsebuje štiri naloge, ki so skupaj vredne 100 točk.
- Za reševanje imate na voljo 90 minut.
- Uporaba literature, interneta ter ostalih medijev NI dovoljena.
- Dovoljena je uporaba zapiskov na dveh A4 listih (napisanih na roko).
- Rešitev vsake naloge pripravite v (ločenem) dokumentu.
- Rešitve (2 skripti v jeziku bash in 2 v C) boste po izteku časa za pisanje v skladu z navodili, ki jih boste prejeli, oddali preko e-učilnice.

Task 1 (20 points)

EN: Write a bash script which does the following:

- Creates a certain number of **`.txt`** files of a specific name and designated number:
  - (2 pts) Asks the user for a file name.
  - (5 pts) Asks the user for two integer numbers, A and B, representing the range. The first number (labelled A) needs to be lower than the second (labelled B). If the first number is not lower than the second, write an error and exit the programme.
  - (5 pts) Creates a certain number of files between ranges A and B in the following format: `filename_2.txt`, `filename_3.txt` etc.
  - (2 pts) Inside the newly generated files, saves the file's name as well.
- (4 pts) Saves the home directory content in a long-form (containing file rights, file sizes, etc.) to the `content.txt` file.
- (2 pts) Displays the contents of the `content.txt` file on the screen.

Example 1

```
user@bash: ./task1.sh
Enter a desired file name:  exam
Enter a range represented as 2 numbers:  1 6

total 28
-rw-rw-r-- 1 user user   0 feb  9 12:08 content.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_1.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_2.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_3.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_4.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_5.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_6.txt
-rw-rw-r-- 1 user user 385 feb  9 12:08 task1.sh
```

Example 2

```
user@bash: ./task1.sh
Enter a desired file name:  exam
enter a range represented as 2 number:  3 1

Error. First number needs to be smaller than the second
```

Naloga 1 (20 točk)

SLO: Napiši bash skripto, ki naredi naslednje:

- Ustvari več datotek s končnico **`.txt`**, z določenim imenom in pripadajočim številom:
  - (2 točki) Vpraša uporabnika za ime datoteke.
  - (5 točk) Vpraša uporabnika za vnos dveh števil A in B, ki predstavljata razpon. Prvo število je označeno z A in drugo z B. A mora biti manjše kot B. V kolikor to ne drži se skripta zaključi z obvestilom in kodo napake.
  - (5 točk) Ustvari več datotek z izbranim imenom in številkami na razponu med A in B. Primer formata: `filename_2.txt`, `filename_3.txt` etc.
  - (2 točki) Znotraj vsake ustvarjene datoteke zapiši ime te datoteke.
- (4 točk) Vsebino domačega direktorija v dolgi obliki (ki vsebuje pravice nad datotekami, velikosti datotek itd.) shrani v datoteko `content.txt`.
- (2 točki) Izpiše vsebino datoteke `content.txt` na zaslon.

Primer 1:

```
user@bash: ./task1.sh
Enter a desired file name:  exam
Enter a range represented as 2 numbers:  1 6

total 28
-rw-rw-r-- 1 user user   0 feb  9 12:08 content.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_1.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_2.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_3.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_4.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_5.txt
-rw-rw-r-- 1 user user   5 feb  9 12:08 exam_6.txt
-rw-rw-r-- 1 user user 385 feb  9 12:08 task1.sh
```

Primer 2:

```
user@bash: ./task1.sh
Enter a desired file name:  exam
enter a range represented as 2 number:  3 1

Error. First number needs to be smaller than the second
```

Task 2 (30 points)

EN: Peter is a software engineer, and he implemented a point of sale (POS) system. It creates log files each day. The name of the file is in the format of `log_<date>.txt`, where the `<date>` is in the format of `YYYY-MM-DD.`

Each log file contains a list of events in the format
`EVENT: <name> | TIME: <time> | USER: <username> | DATA: <data>`.
Write a bash script that extracts a list of all unique events across all log files. (Consider only the name of an event.) Write the extracted **unique** and **alphabetically sorted** event names to a **file** called `event_list.txt`.

Be careful you will not get a guide for this question; points will be allocated as below.
Understanding of the problem (4 points)
Bash script commands (5 points)
Event extraction (7 points)
Alphabetical sorting (6 points)
Output file (4 points)
Following the best practices (4 points)

Example input `log_2023-01-10.txt`

```
EVENT: login | TIME: 2022-03-01 10:15:22 | USER: jdoe | DATA: successful login
EVENT: purchase | TIME: 2022-03-01 10:30:45 | USER: jdoe | DATA: Item #1234 purchased
EVENT: logout | TIME: 2022-03-01 11:00:12 | USER: jdoe | DATA: logout successful
EVENT: login | TIME: 2022-03-02 09:45:33 | USER: jsmith | DATA: successful login
EVENT: logout | TIME: 2022-03-02 11:30:55 | USER: jsmith | DATA: logout successful
EVENT: login | TIME: 2022-03-03 12:15:00 | USER: jdoe | DATA: successful login
EVENT: purchase | TIME: 2022-03-03 12:45:10 | USER: jdoe | DATA: Item #5678 purchased
EVENT: logout | TIME: 2022-03-03 13:00:01 | USER: jdoe | DATA: logout successful
```

Example output (if it only has this log file in the directory):

Here are the unique event names sorted alphabetically.

```
login
logout
purchase
```

Naloga 2 (30 točk)

SLO: Peter je programski inženir in je implementiral sistem prodajnega mesta (POS). Vsak dan ustvari dnevniške datoteke. Ime datoteke je v obliki `log_<datum>.txt`, kjer je `<datum>` v obliki zapisa `LLLL-MM-DD`.
Vsaka dnevniška datoteka vsebuje seznam dogodkov v formatu
`DOGODEK: <ime> | ČAS: <čas> | UPORABNIK: <up. ime> | PODATKI: <podatki>`.

Napišite skripto v jeziku bash, ki ekstrahira seznam vseh edinstvenih dogodkov v vseh dnevniških datotekah. (Upoštevajte samo ime dogodka.). Zapišite ekstrahirana **unikatna** in **abecedno razvrščena** imena dogodkov v **datoteko** z imenom `event_list.txt`.

Bodite previdni, za to vprašanje ne boste dobili dodatnih navodil; način točkovanja:
Razumevanje problema (4 točke)
Uporabljeni ukazi v skripti bash (5 točk)
Ekstrakcija dogodkov (7 točk)
Razvrščanje po abecedi (6 točk)
Izhodna datoteka (4 točke)
Sledenje najboljšim praksam (4 točke)

Primer vhoda `log_2023-01-10.txt`

```
EVENT: login | TIME: 2022-03-01 10:15:22 | USER: jdoe | DATA: successful login
EVENT: purchase | TIME: 2022-03-01 10:30:45 | USER: jdoe | DATA: Item #1234 purchased
EVENT: logout | TIME: 2022-03-01 11:00:12 | USER: jdoe | DATA: logout successful
EVENT: login | TIME: 2022-03-02 09:45:33 | USER: jsmith | DATA: successful login
EVENT: logout | TIME: 2022-03-02 11:30:55 | USER: jsmith | DATA: logout successful
EVENT: login | TIME: 2022-03-03 12:15:00 | USER: jdoe | DATA: successful login
EVENT: purchase | TIME: 2022-03-03 12:45:10 | USER: jdoe | DATA: Item #5678 purchased
EVENT: logout | TIME: 2022-03-03 13:00:01 | USER: jdoe | DATA: logout successful
```

Primer izhoda (če je v direktoriju datoteka z zgornjo vsebino):

Unikatna imena dogodkov, razvrščena po abecedi.

```
login
logout
purchase
```

Task 3 (25 points)

EN: Kaprekar's Constant but with 3 digits

495 is the three-digit Kaprekar's constant. It is derived following the so-called Kaprekar's routine which initially involves 4 digits, the routine is described below:

1. Take any three-digit number, avoid choosing numbers with three identical digits (also called repdigits such as 111 or 999).
2. Arrange the digits into two new three-digit numbers, one where they are arranged in ascending order and another where they are arranged in descending order. If the number contains less than three digits, put some zeroes to the left (called leading zeroes) to maintain 3 digits.
3. Subtract the smaller number from the bigger number.
4. Go back to step 2 and repeat.

Kaprekar's routine for three digits will always reach its fixed point, 495, in at most 6 iterations. Once 495 is reached, the process will continue yielding 954-459 = 495.
For example, consider the number 201. Bigger number: 210 (numbers arranged in descending order). Smaller number: 012 (numbers arranged in ascending order)

Iteration 1: 210-012 = 198*
Iteration 2: 981-189 = 792
Iteration 3: 972-279 = 693
Iteration 4: 963-369 = 594
Iteration 5: 954-459 = 495
Iteration 6: 954-459 = 495
* we used a leading zero for this to make 12 a three digit

Next, consider the number 753. We now have:

Iteration 1: 753-357 = 396
Iteration 2: 963-369 = 594
Iteration 3: 954-459 = 495
Iteration 4: 954-459 = 495

In this case, we achieved Kaprekar's constant after at least three operations.

Your task is to write a C program that will ask the user for a 3-digit input. The program must perform Kaprekar's routine at which it will only stop once it reaches 495 or when it has reached 6 iterations. Your program should be able to reject invalid inputs (repdigits, two-digits, etc). You must print all the steps until you arrive at the constant 495. If possible, print as well the number of iterations it took to arrive at the constant.

Hint: In C, the value "012" when given to an int data type is treated as "12" (without the quotes of course).

Required components to implement:

- 10 points The program performs Kaprekar's routine correctly.
- 05 points The program has input validation (repeat until a valid input is given).
- 05 points The program has descriptive, detailed comments.
- 05 points Use of appropriate naming convention for identifiers and variables.

Some points will be given for solutions that are partially correct.

Naloga 3 (25 točk)

SLO: Kaprekarjeva konstanta, vendar s 3 mestnimi števili

495 je trimestna Kaprekarjeva konstanta. Izpeljana je po tako imenovani Kaprekarjevi rutini, ki j je prvotno vključujevala 4 števke. Kaprekarjeva rutina je opisana spodaj:

1. Vzemite katero koli trimestno število, izogibajte se številom s tremi enakimi ciframi (kot na primer 111 ali 999).
2. Razporedite cifre v dve novi trimestni števili, prvo, kjer so cifre urejene v naraščajočem vrstnem redu, drugo, kjer so urejene v padajočem vrstnem redu. Če število vsebuje manj kot tri cifre, postavite nekaj ničel na levo (imenovane vodilne ničle), da ohranite tri cifre.
3. Od večjega števila odštejte manjše število.
4. Vrnite se na 2. korak in ponovite.

Kaprekarjeva rutina za tri cifre bo vedno dosegla svojo fiksno točko, 495, v največ 6 ponovitvah. Ko je doseženo število 495, se bo postopek nadaljeval in dobili bomo vedno enako število 495 (954-459 = 495).
Vzemimo na primer število 201. Večje število je: 210 (cifre urejene v padajočem vrstnem redu).
Manjše število je: 012 (cifre razvrščene v naraščajočem vrstnem redu)

Ponovitev 1: 210-012 = 198*
Ponovitev 2: 981-189 = 792
Ponovitev 3: 972-279 = 693
Ponovitev 4: 963-369 = 594
Ponovitev 5: 954-459 = 495
Ponovitev 6: 954-459 = 495
* za ta primer smo uporabili vodilno ničlo, da je število 12 postalo trimestno

Vzemimo še število 753. Zdaj imamo:

Ponovitev 1: 753-357 = 396
Ponovitev 2: 963-369 = 594
Ponovitev 3: 954-459 = 495
Ponovitev 4: 954-459 = 495

V tem primeru smo Kaprekarjevo konstanto dosegli po najmanj treh iteracijah.

Vaša naloga je napisati program v jeziku C, ki bo od uporabnika zahteval 3-mestno število. Program mora izvajati Kaprekarjevo rutino in se ustavi, ko je rezultat odštevanja število 495 ali, ko izvede 6 ponovitev. Vaš program mora biti sposoben zavrniti neveljavne vnose (števila s ponovljenimi ciframi (npr. 111), dvomestna števila itd.). Izpisati morate vse korake, dokler ne pridete do konstante 495. Če je mogoče, natisnite tudi število ponovitev, potrebnih za dosego konstante.

Namig: V jeziku C se vrednost »012«, ki je podana v podatkovnem tipu int, obravnava kot »12« (seveda brez narekovajev).

Potrebne komponente za izvedbo:

- 10 točk Program pravilno izvaja Kaprekarjevo rutino.
- 05 točk Program ima validacijo vnosa (ponavljanje računanja, dokler ni podan veljaven vnos).
- 05 točk Program ima opisne, podrobne komentarje.
- 05 točk Uporaba ustrezne konvencije o poimenovanju identifikatorjev in spremenljivk.

Delno pravilne rešitve bodo prejele nekaj točk.

Task 4 (25 points)

EN: 3's Factorial

A factorial of a number *n* is calculated by multiplying all the numbers from 1 to *n*:

$$n!=1*2*3*4*...*n,$$

where *n!* means n-factorial. 7-factorial would be 1*2*3*4*5*6*7 = 5040.

In 3's factorial of a number *n*, we only multiply the numbers from 1 to *n* which are divisible by 3. For example, the 3's factorial of the number 7 would be 3*6=18, because 3 and 6 are the only (positive whole) numbers until 7 which are divisible by 3. Write **a function** that takes as an input a number *n* and outputs the 3's factorial of *n*. Use the function to write a program which takes as input 3 numbers and returns the 3's factorial of each number inputted.

Refer to the examples below to see how the program should function.

Example 1:

Input: 7 6 10
Output: 18 18 162
*Output Explanation : The 3's factorial of 7 is 3*6=18, the 3's factorial of 6 is also 3*6=18, because the numbers divisible by 3 till 7 and 6 are the same, the 3's factorial of 10 is 3*6*9=162.

Example 2:

Input: 3 16 20
Output: 3 29160 524880

Required components to implement:

- 10 points The program runs and prints the correct output.
- 07 points The program uses a function to calculate the 3's factorial.
- 03 points The program uses the correct data type to store the factorial.
- 05 points The program has descriptive, detailed comments and appropriate naming convention for identifiers and variables.

Some points will be given for solutions that are partially correct.

Naloga 4 (25 točk)

SLO: X3 Fakulteta

Fakulteto števila *n* izračunamo tako, da pomnožimo vsa števila od *1* do *n*:

$$n!=1*2*3*4*...*n,$$

kjer n! pomeni n-fakulteta. 7-fakulteta bi bila 1*2*3*4*5*6*7 = 5040.

V X3 fakulteti števila n pomnožimo samo števila od 1 do n, ki so deljiva s 3. Na primer, X3 fakulteta števila 7 bi bilo 3*6=18, ker sta 3 in 6 edini (pozitivni celi) števili manjši od 7, ki sta deljivi s 3. Napišite **funkcijo**, ki vzame kot vhodni podatek število n in izpiše X3 fakulteto števila n. Uporabite to funkcijo pri izdelavi programa v jeziku C, ki vzame kot vhodne podatke 3 števila in vrne X3 fakulteto vsakega vnesenega števila.

Spodnji primeri prikazujejo delovanje programa.

Primer 1:

```
Vhod:  7  10  6
Izhod:18  162  18
```

*Razlaga izhoda: X3 fakulteta števila 7 je 3*6=18, X3 fakulteta števila 6 je prav tako 3*6=18, ker so števila, deljiva s 3 in manjša od 7 ali manjša od 6, enaka, X3 fakulteta števila 10 pa je 3 *6*9=162.

Primer 2:

```
Vhod: 3  16  20
Izhod:3  29160  524880
```

Potrebne komponente za izvedbo:

- 10 točk Program se zažene in natisne pravilen rezultat.
- 07 točk Program uporablja funkcijo za izračun X3 fakultete .
- 03 točke Program uporablja pravilni podatkovni tip za shranjevanje fakultete.
- 05 točk Program ima opisne, podrobne komentarje in ustrezno poimenovanje za identifikatorje in spremenljivke.

Delno pravilne rešitve bodo prejele nekaj točk.