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

Date: / Datum: **9 Jun. 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 (25 points)

EN:
Write a script in **bash** that has at least three functions. (2 pts) The script accepts two integers **`N`** and **`M`** (where **`N < M`**) as a command line arguments and does the following:

- **(4 pts) Function 1**: prints the sum of numbers between **`1`** and **`N`**;
- **(6 pts) Function 2**: prints the sum of numbers between **`N`** and **`M`**, where it uses only **multiples of numbers 3 or 5** to calculate the sum (e.g. `N = 20, M = 30`; numbers to be considered in the sum are: `20, 21, 24, 25, 27, 30`; `sum = 147`);
- **(10 pts) Function 3**: prints all **prime numbers** between **`N`** and **`M`**.

(3 pts) It should also check that the user has entered positive integers. If the user entered something else, print out the error.

Example 1:

```
user@bash: ./task1.sh 20 30
The sum of numbers between 1 and 20 is 210.
The sum of the multiples of 3 or 5 between 20 and 30 is 147.
Prime numbers between 20 and 30 are: 23, 29
```

Example 2:

```
user@bash: ./task1.sh N 30
Error: N is not an integer. Please provide a positive integer.
```

Example 3:

```
user@bash: ./task1.sh -4 30
Error: -4 is not a positive integer. Please provide a positive
integer.
```

Naloga 1 (25 točk)

SLO:
Napišite skripto v **bash**u, ki ima vsaj tri funkcije. (2 t) Skripta sprejme dve celi števili **N** in **M** (kjer **N < M**) kot argumenta ukazni vrstici in naredi naslednje:

- **(4 t) Funkcija 1**: izpiše vsoto števil med **1** in **N**;
- **(6 t) Funkcija 2**: izpiše vsoto števil med **N** in **M** , kjer za izračun vsote uporabi le **večkratnike števil 3 ali 5** (npr. `N = 20, M = 30`; števila, ki se upoštevajo v vsoti: `20, 21, 24, 25, 27, 30; vsota = 147`);
- **(10 t) Funkcija 3**: izpiše vsa **praštevila** med **N** in **M**.

(3 t) Prav tako preveri, da je uporabnik podal celi števili, ki morata biti pozitivni. Če karkoli od tega ne velja, izpiši napako.

Primer 1:

```
user@bash: ./task1.sh 20 30
The sum of numbers between 1 and 20 is 210.
The sum of the multiples of 3 or 5 between 20 and 30 is 147.
Prime numbers between 20 and 30 are: 23, 29
```

Primer 2:

```
user@bash: ./task1.sh N 30
Error: N is not an integer. Please provide a positive integer.
```

Primer 3:

```
user@bash: ./task1.sh -4 30
Error: -4 is not a positive integer. Please provide a positive
integer.
```

Task 2 (25 points)

EN: John is a system administrator responsible for managing a server infrastructure. He wants to monitor the disk usage of a specific directory on the server and receive an email notification when the disk usage exceeds a certain threshold.

Write a bash script that performs the following tasks:

- (2 pts) Accepts user input for the directory path to monitor and the disk usage threshold (in percentage).
  - User path should be accepted while the script is running (not as a command line argument)
  - Provided path should be validated (display errors appropriately)
- (5 pts) Retrieves the current disk usage of the given directory.
  - This task should be implemented in a dedicated function
  - Script should check the disk usage once every **N minutes**
- (2 pts) Compares the disk usage with the threshold value.
- (7 pts) If the disk usage exceeds the threshold, the script sends an email notification to John's email address with the subject "Disk Usage Alert" and the message containing the directory path and the current disk usage percentage.
  - Should implement a dedicated function for sending an email.
  - Subject “Disk Usage Alert” should be passed as an argument to the function.
  - Assume the email server is already configured and you can send an email with the following command, help is provided.
    - `Mail`

Argument	**Description**
-s	Subject
-t	Receiver’s email address
-m	Message body

- (5 pts) If the disk usage is below or equal to the threshold, displays a message stating that the disk usage is within the acceptable limits.
  - Message should be appended into a log file `usage.log`
- (4 pts) Ensure that your script handles error scenarios, such as incorrect directory paths or non-numeric threshold values. You may assume that the necessary email configuration is already set up on the server.

Example 1

```
user@bash: ./task2.sh

Enter the directory path to monitor: /var/www/html
How frequently should script check disk usage (in mins): 9
Enter the disk usage threshold (in percentage): 90

Checking disk usage...

Disk usage in /var/www/html: 92%

Sending email notification to John (john@example.com)...

Email sent successfully.
```

Naloga 2 (25 točk)

SLO: John je sistemski skrbnik, odgovoren za upravljanje strežniške infrastrukture. John želi spremljati, koliko diska na strežniku porabi določena mapa in prejeti obvestilo preko e-pošte, ko poraba diska preseže določen prag.

Napišite skripto bash, ki izvaja naslednje naloge:

- (2 pts) Sprejme uporabniška vnosa za pot do željene mape za nadzor, ter prag dovoljene porabe diska (v odstotkih).
    - Pot do mape sprejme med izvajanjem skripte (ne kot parameter v ukazni vrstici)
    - Preverite, da je navedena pot ustrezna (prikaži napako, če pot ni ustrezna)
- (5 pts) Preveri trenutno porabo diska za dano mapo.
    - Ta del mora biti izveden v namenski funkciji.
    - Skripta naj preveri porabo diska vsakih **N minut**.
- (2 pts) Primerja porabo diska z zgoraj določenim pragom.
- (7 pts) Če uporaba diska preseže prag, skripta pošlje e-poštno obvestilo na Johnov e-poštni naslov z zadevo »Opozorilo o porabi diska« in sporočilom, ki vsebuje pot do mape in trenutni odstotek porabe diska.
    - Implementiraj namensko funkcijo za pošiljanje e-pošte.
    - Zadevo sporočila »Opozorilo o porabi diska« posredujete kot argument funkciji.
    - Predpostavimo, da je e-poštni strežnik že konfiguriran in lahko pošljete e-pošto z naslednjim ukazom, pomoč je na voljo.
        - `Mail`

Parameter	**Opis**
-s	Zadeva
-t	Prejemnikov e-poštni naslov
-m	Vsebina sporočila

- (5 pts) Če je poraba diska nižja ali enaka določenemu pragu, skripta prikaže sporočilo v Terminal, da je poraba diska znotraj sprejemljivih meja.
    - Sporočilo je treba dodati v dnevniško datoteko `usage.log`
- (4 pts) Zagotovite, da vaša skripta obravnava scenarije napak, kot so nepravilne poti do mape ali neštevilske vrednosti praga. Predvidevate lahko, da je potrebna konfiguracija e-pošte že nastavljena na strežniku.

Primer 1

```
user@bash: ./task2.sh

Vstavi pot do željenje mape: /var/www/html
Kako pogosto naj se preveri poraba (v minutah): 9
Vstavi željeni prag porabe diska (v odstotkih): 90

Preverjam porabo diska...

Poraba diska za /var/www/html: 92%

Pošiljanje e-pošte (john@example.com)...

Email uspešno poslan.
```

Task 3 (25 points)

EN: Armstrong Numbers

Write a program that determines whether a given number is an Armstrong number or not. An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits. Your program **should implement a function** that takes an integer as input and returns *true* if the number is an Armstrong number, and *false* otherwise.

Sample valid inputs and outputs:

```
Input 1: 153
Output 1: Is Armstrong number? Yes
```

Explanation: 153 has three digits. So the raised power is 3. Add the sum of the powers of each digit. 1^3 + 5^3 + 3^3 = 153.

```
Input 2: 370
Output 2: Is Armstrong number? Yes
```

```
Input 3: 123
Output 3: Is Armstrong number? No
```

Grading breakdown:

1. Functionality (15 points):
   - Implementing the function correctly: 10 points
   - Handling non-negative integer inputs: 2 points
   - Calculating the sum of digits raised to the power of the number of digits: 3 points
2. Code structure and naming convention (5 points):
   - Proper use of functions: 2 points
   - Meaningful variable and function names: 2 points
   - Consistent indentation and formatting: 1 point
3. Comments and documentation (5 points):
   - Clear and concise comments explaining the code's purpose: 2 points
   - Function and parameter descriptions: 2 points
   - Overall readability and documentation quality: 1 point

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

Naloga 3 (25 točk)

SLO: Armstrongova števila

Napišite program, ki ugotovi, ali je dano število Armstrongovo število ali ne. Armstrongovo število je število, ki je enako vsoti svojih števk, ki so potencirane s številom števk. Vaš **program naj vsebuje funkcijo**, ki kot vhod sprejme celo število in vrne *true*, če je število Armstrongovo, in *false* v nasprotnem primeru.

Primeri veljavnih vhodnih in izhodnih podatkov:

```
Vhod 1: 153
Rezultat 1: Je Armstrongovo število? Da
```

Razlaga 1: 153 ima tri števke. Potenca bo torej 3. Nato seštejemo vse potencirane števke, torej dobimo: 1^3 + 5^3 + 3^3 = 153.

```
Vnos 2: 370
Rezultat 2: Ali je Armstrongovo število? Da

Vnos 3: 123
Rezultat 3: Ali je Armstrongovo število? Ne
```

Točkovanje:

1. Funkcionalnost (15 točk):
   - Pravilna implementacija funkcije: 10 točk
   - Ravnanje z vhodnimi nenegativnimi celimi števili: 2 točki
   - Izračun vsote potenciranih števk: 3 točke
2. Struktura kode in način poimenovanja (5 točk):
   - Pravilna uporaba funkcij: 2 točki
   - Smiselna imena spremenljivk in funkcij: 2 točki
   - Primerni ter konsistentni zamiki in oblikovanje: 1 točka
3. Komentarji in dokumentacija (5 točk):
   - Jasni in jedrnati komentarji, ki pojasnjujejo namen kode: 2 točki
   - Opisi funkcij in parametrov: 2 točki
   - Splošna berljivost in kakovost dokumentacije: 1 točka

Delno pravilne rešitve bodo prejele nekaj točk.

Task 4 (25 points)

EN: Ticket reservation system

You are tasked to create a simple program that emulates a cinema ticket reservation system for a cinema with a single screen and 100 seats organised in a 10x10 grid.

The seating arrangement should be presented to the user as a 10x10 grid. Each seat is identified by its position in the grid (e.g., the first seat is at position 0,0, and the last seat is at position 9,9). For visual representation, the program should use 'A' for available seats and '_' for reserved seats.

The program should begin by displaying a greeting message to the user and the current seating map, where all seats are initially available. It should then prompt the user to enter a seat number to reserve. After the user provides a valid seat number, the program should reserve that seat and display the updated seating map. If the seat is already reserved, the program should inform the user and prompt them to choose a different seat.

After reserving a seat and showing the updated seating map, the program should ask the user whether they want to reserve another seat. If the user decides to continue, the program should repeat the reservation process. Otherwise, it should end with a closing message.

Example 1:

```
Welcome to the Cinema Reservation System!

Current Seating Map:

A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

Please enter the seat number you wish to reserve (1-100): 35

Seat 35 has been successfully reserved!

Updated Seating Map:

A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A _ A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

Would you like to reserve another seat? (yes/no): no

Thank you for using the C Cinema Reservation System! Goodbye.
```

Required components to implement:

- 5 points The seating array is correctly initialized and displayed at the start.
- 5 points The seat reservation function correctly reserves available seats, handles already reserved and invalid seats.
- 5 points The program flow is logical and user-friendly, with clear prompts and effective handling of user responses.
- 5 points The seating map accurately reflects the current reservation status, being correctly updated and displayed after each reservation.
- 5 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: Rezervacijski sistem v gledališču

Vaša naloga je ustvariti program, ki posnema sistem rezervacij gledaliških vstopnic v gledališču z enim odrom in 100 sedeži, postavljenimi v mreži 10x10.

Razporeditev sedežev naj bo uporabniku prikazana kot mreža 10x10. Vsak sedež je identificiran s svojo pozicijo v mreži (npr., prvi sedež je na položaju 0,0, in zadnji sedež je na položaju 9,9). Za vizualno predstavitev naj program uporablja 'A' za razpoložljive sedeže in '_' za rezervirane sedeže.

Program naj se začne z uvodnim sporočilom uporabniku in prikaz trenutne sedežne postavitve, kjer so na začetku vsi sedeži prosti. Nato naj uporabnika pozove, naj vnese številko sedeža, ki ga želi rezervirati. Ko uporabnik navede veljavno številko sedeža, naj program rezervira ta sedež in prikaže posodobljeno sedežno mrežo. Če je sedež že rezerviran, naj program uporabnika obvesti in ga pozove, naj izbere drug sedež. Po rezervaciji sedeža in prikazovanju posodobljene sedežne mreže naj program vpraša uporabnika, ali želi rezervirati še en sedež. Če se uporabnik odloči nadaljevati, naj program ponovi postopek rezervacije. Sicer naj se zaključi z zaključnim sporočilom.

Primer 1:

```
Dobrodošli v sistem rezervacij vstopnic za gledalisce!

Trenutni sedezni red:
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

Prosimo, vnesite stevilko sedeza, ki ga zelite rezervirati (1-100): 35

Sedez 35 je bil uspesno rezerviran!

Posodobljen sedezni red:
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A _ A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

Zelite nadeljevati z rezervacijami? (da/ne): ne

Hvala, ker ste uporabili naš sistem za rezervacijo vstopnic! Nasvidenje.
```

Potrebne komponente za izvedbo:

- 5 točk Seznam sedežev je na začetku pravilno inicializiran in prikazan.
- 5 točk Rezervacija sedeža deluje pravilno in preveri že rezervirane in neveljavne sedeže.
- 5 točke Programski tok je logičen in prijazen do uporabnika, z jasnimi pozivi in učinkovitim obvladovanjem odzivov uporabnikov.
- 5 točk Sedežna mreža pravilno odraža trenutni status rezervacij, se pravilno posodobi in prikaže po vsaki rezervaciji.
- 5 točk Program ima opisne, podrobne komentarje in ustrezno poimenovanje funkcij in spremenljivk.

Delno pravilne rešitve bodo prejele nekaj točk.