Zadanie nr 1

```
#!/bin/bash

#
================================================================
================================================================
========================
# Zad.1. (2 punkty - na zajęciach)
# Wykorzystaj date do wyświetlania różnych komunikatów powitalnych w
zależności czy aktualnie jest dzień roboczy
# czy świąteczny (dla uproszczenia: dzień tygodnia czy weekend).
#
# Uwaga: Porównywanie napisów wyświetlanych przez program date (i inne
programy) jest zależne od języka i lokalizacji.
#
# Sprawdź możliwość ustawienia zmiennych lokalizacji LC_* (np. LC_ALL=C) i
ich wpływ na postać komunikatu.
# Która/które z nich sterują postacią wyświetlanej daty? Ustawiając tę zmienną
w skrypcie na wartość C (lub POSIX),
# wymuszamy lokalizację kanoniczną, co ułatwia sprawdzanie wartości daty, i
uniezależnia działanie skryptu
# od lokalizacji (będzie on poprawnie sprawdzał warunek również np. w
Japonii).
#
================================================================
================================================================
========================

# pobranie aktualnego dnia tygodnia
# date +%u <- zwraca numer dnia tygodnia, gdzie 1 to poniedziałek, a 7 to
niedziela.
day_of_week=$(date +%u)

# Sprawdzenie czy jest dzień roboczy (1-5 to dni robocze: Pon-Pt)
# -gt – większe niż (greater than)
# -lt – mniejsze niż (less than)
# -eq – równe (equal)
# -ne – różne (not equal)
if [ "$day_of_week" -ge 1 ] && [ "$day_of_week" -le 5 ]; then
    echo "Witaj, dzisiaj jest dzień roboczy..."
else
    # Jeśli dzień to sobota (6) lub niedziela (7)
    echo "Witaj, dzisiaj jest weekend!"
fi
```

Zadanie nr 2

```
#!/bin/bash

#
================================================================
================================================================
========================
#Zad.2. (2 punkty - na zajęciach)
#Napisz skrypt o nazwie policz.sh, który policzy i wyświetli liczbę plików w
katalogu bieżącym. WSKAZÓWKA:
#opcja -1 do programu ls powoduje wyświetlanie po jednym pliku w linijce, a
program wc można wywołać z opcją -l
#powodującą policzenie tylko liczby linijek.
#
#Następnie zmodyfikuj ten skrypt aby rozpoznawał on argument opcjonalny -a,
podobnie jak program ls. To znaczy,
#z argumentem -a skrypt obliczy liczbę wszystkich plików w katalogu
bieżącym, włącznie z plikami ,,ukrytymi''
#(o nazwach zaczynających się od kropki), a bez -a pominie te pliki.
#
================================================================
================================================================
========================

# check for an "-a"
# wc -l <- ounts the number of lines
if [ "$1" == "-a" ]; then
    # count all files, hidden too (ls -A -1)
    echo "Pliki: "
    ls -A -1
    echo $'\n'
   files_ctr=$(ls -A -1 | wc -l)
else
    # count only visible files (ls -1)
    echo "Pliki: "
    ls -1
    echo $'\n'
   files_ctr=$(ls -1 | wc -l)
fi

echo "Liczba plików: $files_ctr"
```

Zadanie nr 3

```
#!/bin/bash

#
==========================================================
==========================================================
==========================
#Zad.3. (2 punkty - na zajęciach, lub 1 punkt - w domu)
#Wykorzystując instrukcję pętli for napisz skrypt, który dla dowolnej liczby
argumentów pozycyjnych wyświetli informacje
#o każdym z nich traktowanym jako nazwa pliku. Jeśli dany argument określa
istniejący plik, to powinien
#się pojawić komunikat "plik istnieje", i analogicznie, jeśli nie istnieje plik o
podanej nazwie. W przypadku gdyby
#podany argument określał katalog, należy wyświetlić "katalog", ale jeśli jest
to katalog pusty (nie zawiera żadnych
#plików, ani podkatalogów, z wyjątkiem . i ..), to należy wyświetlić "katalog
pusty".
#
==========================================================
==========================================================
==========================

# for loop for every file from arguments
for arg in "$@"; do
    # -d <- check if arg is a directory
    if [ -d "$arg" ]; then
        # ls -A <- show files in a directory
        if [ "$(ls -A "$arg")" ]; then
            echo "$arg - katalog"
        else
            echo "$arg - katalog pusty"
        fi
    # case when agr is a file
    # -f <- check if arg is a file
    elif [ -f "$arg" ]; then
        echo "$arg - plik istnieje"
    # in case when there isn't such an arg
    else
        echo "$arg - plik/katalog nie istnieje"
    fi
done
```

Zadanie nr 4

```
#!/bin/bash

#
================================================================
================================================================
========================
# Zad.4. (3 punkty - na zajęciach, lub 2 punkty - w domu)
# Ćwiczenia z programem find: napisz skrypt znajdujący programem find
wszystkie found_files z podanego katalogu, o podanym
# rozszerzeniu (końcówce w nazwie), które były modyfikowane w ciągu
ostatnich N dni i tworzący z nich archiwum tar
# o zadanej nazwie. Mamy tu cztery parametry: nazwa katalogu, rozszerzenie
nazwy plików, liczbę N i nazwę archiwum.
# Parametry będą przekazane do skryptu jako argumenty pozycyjne, to znaczy
skrypt zawsze będzie wywołany z czterema
# parametrami w podanej kolejności.
#
# Uwaga: spróbuj tak napisać skrypt, aby uwzględniał tylko found_files, do
których użytkownik go wywołujący ma dostęp
# przynajmniej typu read, to znaczy by nie generował błędów dla plików
niedostępnych, tylko pomijał te found_files.
#
================================================================
================================================================
========================

# exit program while user prompts wrong number of arguments
# $# <- how many arg
# -ne <- not equal
if [ "$#" -ne 4 ]; then
    echo "Zła ilość argumentów (nazwa katalogu, rozszerzenie nazwy plików,
liczbę N i nazwę archiwum)."
    exit 1
fi

# save args to virables
path="$1"
extension="$2"
time="$3"
tar_name="$4"

# saving matchig files to "found_files"
# -type f <- looks only for files
# -name "*.$extension" <- looks for files ened with "ectension"
# -mtime <- how long ago was this file modified
# -perm /u=r <- files with read permision to avoid errors
# -exec basename {} \; <- saves only files names exept whole path
found_files=$(find "$path" -type f -name "*.$extension" -mtime "-$time" -
perm /u=r -exec basename {} \;)
```

```
# check if program found any files
# -z <- zero-length
if [ -z "$found_files" ]; then
    echo "Nie znaleziono żadnych plików."
    exit 1
fi

# make tar archive
tar -czf "$tar_name.tar" $found_files
```

Zadanie nr 5

```
#!/bin/bash

# ================================================================================
# Zad.5. (3 punkty - na zajęciach, lub 2 punkty - w domu)
# W tym ćwiczeniu chcemy wykorzystać instrukcję pętli logicznej while sterowanej strumieniem danych. To znaczy, chcemy
# czytać strumień danych wiersz po wierszu, wykonując jakieś operacje, i zatrzymać wykonywanie tego skryptu
# po napotkaniu końca strumienia danych. Ten schemat można zapisać tak:
#
# while read x y z
# do
# ... # dowolne operacje na danych wczytanych z kolejnych wierszy pliku do zmiennych x y z
# done < plik
#
# Jako przykład zastosowania takiego schematu napisz skrypt do wysyłania emaila o ustalonej treści do wszystkich
# adresów zapisanych w kolejnych wierszach w pliku.
#
# Znajdź program klienta poczty elektronicznej pozwalającego wysyłać maile z wiersza poleceń w trybie nieinterakcyjnym
# (np. mail/mailx, pine/alpine, mutt, itp). Przeczytaj jego opis i opracuj wyrażenie shella pozwalającego automatycznie
# wysłać maila o jakiejś ustalonej treści. Następnie użyj tego wyrażenia w pętli do wysyłania maila automatycznie
# do kolejnych adresów.
#
# Uwaga: emaile muszą być rozesłane indywidualnie do wszystkich adresatów. Niedopuszczalne jest wysłanie jednego emaila
# do listy zawierającej wszystkie adresy.
#
# ================================================================================
```

```
# empty mail list
mail_list=""

# function to sending emails
send_email() {
    local email_adress="$1"
    local subject="$2"
    local message="$3"

    echo -e "$message" | mail -s "$subject" "$email_adress"
}

echo "Podaj mail'a (wpisz 'EOF' aby zakończyć):"
while true; do
    read temp
    if [ "$temp" == "EOF" ]; then
        break
    else
        # adding mails to list (every mail to a new line)
        mail_list+="$temp"$'\n'
    fi
done

# check if there are any mails
if [ -z "$mail_list" ]; then
    echo "Brak adresów mail"
    exit 1
else
    echo "Lista maili:"
    echo "$mail_list"
fi

# mail content
temat="SPAM"
tresc="Skrypt_05 działa"

# sending messages to saved mails addresses
echo "$mail_list" | while read -r emial_adress; do
    send_email "$email_adress" "$temat" "$tresc"
done
```