Systemy operacyjne
Laboratorium 6 (Bash - powtórka)

Tutaj znajduje się prezentacja z przykładami użycia języka skryptowego **BASH**.

Zadanie 1 (0.1 pkt)**.** Napisz i uruchom skrypt wczytujący i wyświetlający łańcuch (napis). Przykładowe użycie programu:

```
./zad1.sh
Podaj napis:
ala ma kota
Napisales "ala␣ma␣kota"
```

Zwróć uwagę, że skrypt wypisał wczytany napis w cudzysłowiu!

Zadanie 2 (0.2 pkt)**.** Napisz i uruchom skrypt który będzie pobierał liczbę naturalną n, a następnie pokaże kolejne n potęg postaci $2^0, 2^1, \cdots, 2^n$.
Przykładowe wywołania programu:

```
Podaj liczbe:
8
2 4 8 16 32 64 128 256
Podaj liczbe:
mama2
Nie liczba! Koncze program!
```

Zwróć uwagę że ma być wczytywana liczba. Zwróć uwagę na wyjście skryptu: wszystkie liczby zapisane są w jednej linii.
Wskazówka: Można skorzystać z programu **bc**, w którym jest zdefiniowane potęgowanie (operator ^).

Zadanie 3 (0.2 pkt)**.** Napisz i uruchom skrypt, który:

a) Otrzymuje trzy argumenty wywołania i kolejno je wyświetla.

b) Otrzymuje dowolną liczbę argumentów i wyświetla je wszystkie.

Przykładowe wywołania skryptu:

```
./zad3a.sh 1 2
Mialo byc dokladnie 3 argumenty! Koncze program!
./zad3a.sh 1 2 mama
1 2 mama
./zad3a.sh 1 2 mama tata
Mialo byc dokladnie 3 argumenty! Koncze program!
./zad3b.sh 1 2 mama tata
1 2 mama tata
./zad3b.sh
Nie podales zadnych argumentow! Koncze program!
```

W przypadku uruchamiania skryptów, (podanie w linii poleceń nazwy skryptu i ewentualnych argumentów wywołania), tworzony jest nowy proces (potomny względem bieżącego interpretera), który inicjuje wykonywanie skryptu. Jest też jednak możliwe uruchomienie skryptu w ramach procesu bieżącego interpretera.

Zadanie 4 (0.2 pkt)**.** Poszukaj jak można uruchomić skrypt w ramach procesu bieżącego interpretera. Utwórz skrypt, który tworzy nową zmienną środowiskową i nadaje je jakąś wartość. Później uruchom skrypt:

a) W podpowłoce.

b) W bieżącej powłoce.

Jaka jest różnica? Odpowiedź uzasadnij.
UWAGA! Skrypt nie powinien zawierać na końcu exit 0. Za każdym razem po wywołaniu skryptu należy sprawdzić czy utworzona zmienna nadal istnieje, a jeśli tak, to usunąć ją.

Zadanie 5 (0.4 pkt)**.** Napisz skrypt, który zadeklaruje tablicę 10-elementową. Wypełni ją dowolnymi liczbami naturalnymi, a następnie wyświetli na ekranie tę tablicę podniesioną do kwadratu. Przykładowe wywołanie skryptu:

```
./zad5.sh
Orginalna tablica:
16 2 5 19 14 20 20 3 12 15
Tablica do kwadratu:
256 4 25 361 196 400 400 9 144 225
```

Każdy program/skrypt/polecenie zwraca tzw. **exit code**. Exit code jest jednym ze sposobów sprawdzenia czy program/skrypt/polecenie zostało wykonane pomyślnie czy nie. Domyślnie według standardu **POSIX** (Portable Operating System Interface) powodzenie ma exit code równy 0, zaś 1 oznacza się niepowodzenie programu/skryptu/polecenia.

Zadanie 6 (0.1 pkt)**.** Poszukaj jak można sprawdzić w bash'u **exit code** poprzedniego polecenia. Sprawdź jaki będzie **exit code** po wykonaniu po kolei każdego poniższego polecenia. Poszukaj znaczenie każdej wartości **exit code** w internecie.

```
ls -l
echo ;;
$/usr/bin
foo
read (nastepnie wcisnac kombinacje ctrl+c)
let "zmienna=␣40/0"
```

ZADANIA DOMOWE

Zadanie 7 (0.3 pkt)**.** W bash'u można tak jak w każdym języku programowania pisać funkcję. Napisz skrypt, który będzie wywoływany następująco:

```
./skrypt 1 4
```

a na wyjściu wyświetli $nwd(a, b)$ (największy wspólny dzielnik liczb a i b). Skrypt powinien zawierać osobno funkcje, która oblicza nwd.
Przykładowe wywołania skryptu:

```
./zad7.sh 32 517
1
./zad7.sh 32 64
32
./zad7.sh 32 mama
```

```
Argumenty nie sa liczbami! Koncze program!
./zad7.sh
Mialo byc dokladnie dwa argumenty! Koncze program
./zad7.sh 1
Mialo byc dokladnie dwa argumenty! Koncze program
```

Zadanie 8 (0.5 pkt)**.** W pliku **zadanie8.zip** znajduje się program napisany w języku **ANSI C** oraz przykładowe dane testowe tego programu. Skompiluj program a następnie napisz skrypt, który uruchomi ten program dla wszystkich danych testowych zapisanych w katalogu **Input** i zapisze wyniki działania tego programu w folderze **Results**.
UWAGA! Należy utworzyć katalog **Results**. Każdy wynik działania programu należy zapisać w innym pliku np: result1.txt, result2.txt itd. Pliki testowe są nazywane odpowiednio: test.txt, test1.txt, test2.txt. Skrypt powinien w argumencie pobierać jak uruchamia się dany program. Przykładowe wywołanie skryptu:

```
./zad8.sh
Bledne wywolanie! Koncze program!
./zad8.sh ./a.out # Skrypt zadzialal -> Wyniki sa w folderze Results
```
