Programowanie I R

Zadania – seria 5.

Pliki.

Zadanie 1. `comments` – Komentarze w pliku tekstowym.

Napisz program `comments` służący do usuwania komentarzy z pliku tekstowego. Program powinien przyjmować trzy argumenty wywołania: pierwszy ma być pewnym znakiem, zaś pozostałe dwa – nazwami plików. Zadaniem programu jest przepisanie zawartości pierwszego z tych plików do drugiego z nich z pominięciem linii zaczynających się znakiem przekazanym programowi jako pierwszy argument wywołania.

Przykładowe wykonanie
Zawartość pliku wejściowego `in.txt`*:*

```
pierwsza linia
?komentarz
druga linia
```

Wywołanie:

```
python comments.py ? in.txt out.txt
```

Zawartość pliku wyjściowego `out.txt`*:*

```
pierwsza linia
druga linia
```

Zadanie 2. `bill` – Obliczanie kwoty do zapłaty.

Napisz program `bill`, którego zadaniem będzie analiza zapisanego w prostej formie rachunku i obliczenie całkowitej należności. Program ma przyjmować jako argument wywołania nazwę pliku tekstowego. Plik ten powinien zawierać w kolejnych linach nazwy zakupionych produktów, składające się z dowolnych znaków, w tym spacji, oraz ich ceny, umieszczone zawsze na końcu linii, po spacji. Ilość zakupionych produktów nie jest znana z góry. Zadaniem programu jest zsumowanie cen wszystkich produktów i wypisanie obliczonej w ten sposób całkowitej kwoty do zapłaty na standardowe wyjście.

Przykładowe wykonanie
Zawartość pliku wejściowego `grocery.txt`*:*

```
Serniczek z rodzynkami 19.90
Czekolada 90% 5.60
Pomidor 3.14
```

Wywołanie:

```
python bill.py grocery.txt
```

Wyjście:

```
28.64
```

Zadanie 3. `textcalc` – Kalkulator tekstowy.

Napisz funkcję `TextCalc`, która przyjmuje jako argument łańcuch tekstowy przedstawiający działanie matematyczne, posiadający następującą strukturę:

- ciąg znaków dający się zinterpretować jako liczba rzeczywista (składający się z cyfr i ewentualnie jednej kropki),
- spacja,
- jeden z następujących znaków: `+`, `-`, `*`, `/`, `%` , `^`,
- spacja,
- ciąg znaków dający się zinterpretować jako liczba rzeczywista (składający się z cyfr i ewentualnie jednej kropki).

Wymienione znaki symbolizują operacje matematyczne, odpowiednio: dodawanie, odejmowanie, mnożenie, dzielenie, obliczanie reszty z dzielenia oraz potęgowanie. Funkcja `TextCalc` powinna zwracać liczbę zmiennoprzecinkową reprezentującą wynik operacji matematycznej powiązanej z danym znakiem, wykonanej na zadanych liczbach. Na przykład dla argumentu `"1.3 + 2.5"` funkcja powinna zwrócić wartość `3.8`.

Korzystając z tej funkcji napisz program `textcalc` będący prostym kalkulatorem tekstowym. Program powinien przyjmować jako argumenty wywołania nazwy dwóch plików tekstowych. Pierwszy z tych plików ma zawierać ciąg działań matematycznych zapisanych w postaci opisanej powyżej, takiej samej, jak w przypadku argumentu funkcji `TextCalc`. Każde działanie ma być zapisane w osobnej linii, ilość działań nie jest znana z góry. Zadaniem programu jest przepisanie zawartości tego pliku do drugiego z plików, przy czym na końcu każdej linii program powinien dopisać spację, znak równości, kolejną spację i wynik działania zapisanego w tej linii.

Przykładowe wykonanie
Zawartość pliku wejściowego `in.txt`*:*

```
3.67 + 2
3.2 * 4
3.14 / 2.72
```

Wywołanie:

```
python textcalc.py in.txt out.txt
```

Zawartość pliku wyjściowego `out.txt`*:*

```
3.67 + 2 = 5.67
3.2 * 4 = 12.8
3.14 / 2.72 = 1.15
```

Zadanie 4. `enigma` – Szyfrowanie z użyciem XOR.

Jedną z metod szyfrowania plików tekstowych jest zastąpienie każdego znaku bitową różnicą symetryczną (*XOR*) jego kodu ASCII z zadaną jednobajtową liczbą zwaną kluczem. Odszyfrowanie tekstu polega na ponownym obliczeniu bitowej różnicy symetrycznej zaszyfrowanych kodów z tym samym kluczem.

Napisz program `enigma` szyfrujący i deszyfrujący pliki tekstowe omówioną metodą. Program powinien przyjmować cztery argumenty wywołania. Pierwszy z nich to `-e` dla szyfrowania lub `-d` dla deszyfrowania, drugi – jednobajtowa liczba całkowita będąca kluczem, trzeci jest nazwą pliku z tekstem do zaszyfrowania/odszyfrowania, zaś czwarty – nazwą pliku, w którym ma zostać zapisany zaszyfrowany/odszyfrowany tekst. Szyfrowanie ma się odbywać znak po znaku, włączając znaki białe. Plik z zaszyfrowanym tekstem powinien zawierać oddzielone spacjami liczby – każda z nich jest bitową różnicą symetryczną kodu ASCII zaszyfrowanego znaku i klucza.

Przykładowe wykonanie 1.
Zawartość pliku wejściowego `in.txt`*:*

```
Non intratur in veritatem, nisi per caritatem.
```

Wywołanie:

```
python enigma.py -e 7 in.txt out.txt
```

Zawartość pliku wyjściowego `out.txt`*:*

```
73 104 105 39 110 105 115 117 102 115 114 117 39 110 105 39 113 98
117 110 115 102 115 98 106 43 39 105 110 116 110 39 119 98 117 39
100 102 117 110 115 102 115 98 106 41
```

Przykładowe wykonanie 2.

Zawartość pliku wejściowego `in.txt`*:*

```
73 104 105 39 110 105 115 117 102 115 114 117 39 110 105 39 113 98
117 110 115 102 115 98 106 43 39 105 110 116 110 39 119 98 117 39
100 102 117 110 115 102 115 98 106 41
```

Wywołanie:

```
python enigma.py -d 7 in.txt out.txt
```

Zawartość pliku wyjściowego `out.txt`*:*

```
Non intratur in veritatem, nisi per caritatem.
```

Opracowanie: Bartłomiej Zglinicki.