Zajęcia 5 – łańcuchy znaków (ciąg dalszy) i funkcje

1. Napisz funkcję, która zwraca wartość silni dla podanej liczby n. Funkcja powinna być napisana w dwóch wersjach: iteracyjnej i rekurencyjnej.

2. Napisz funkcję, która zwraca wartość n-tego wyrazu ciągu Fibonacciego. Funkcja powinna być napisana w dwóch wersjach: iteracyjnej i rekurencyjnej.

   Ciąg Fibonacciego dany jest wzorem:

$$F_n = \begin{cases} 1 & \text{gdy } n = 0\ , \\ 2 & \text{gdy } n = 1\ , \\ F_{n-2} + F_{n-1} & \text{gdy } n > 1\ . \end{cases}$$

3. Zdefiniować funkcję `int strpos(String text, char z)`, która zwraca indeks na którym znajduje się znak `z` (drugi parametr) w podanym łańcuchu `text`. Jeżeli znak `z` nie występuje w łańcuchu, to wynikiem funkcji powinno być -1. Uwaga - pozycje znaków numerujemy począwszy od 0.

4. Napisać funkcję `String flipCase(String text)`, która zamieni małe litery na duże i odwrotnie w łańcuchu podanym jako parametr. Wynikiem ma być łańcuch znaków zawierający kopię łańcucha po zmianie wielkości liter.

5. Zdefiniować funkcję `boolean startsWith(String str1, String str2)`, która sprawdza, czy łańcuch `str2` jest prefiksem łańcucha `str1`.

   Przykłady:

```
startsWith("Alibaba", "Ali") - wynik true, ponieważ wyraz
                               "Alibaba" zaczyna się wyrazem "Ali".
startsWith("Alibaba", "Alibaba") - wynik true, ponieważ wyraz jest
                                   zawsze swoim prefiksem.
startsWith("Kot", "Pies") - wynik false, ponieważ wyraz "Pies"
                            nie jest prefiksem wyrazu "Kot"
```

6. Zdefiniować funkcję `int strToInt(String str)`, która zamienia liczbę całkowitą zapisaną w postaci łańcucha na liczbę całkowitą typu int. Funkcja powinna przerywać konwersję w momencie napotkania pierwszego znaku nie należącego do zapisu liczby, zatem np. dla `strToInt("-13krowa")` wynikiem powinno być -13.

   Pozostałe przykłady:

```
strToInt("+12") - wynik 12
strToInt("0001") - wynik 1
strToInt("991-234-23") - wynik 991
strToInt("+zonk") - wynik 0
strToInt("") - wynik 0
strToInt("-12e5") - wynik -12*10^5 = -120000
strToInt("-12e-5") - wynik -12, tylko dodatnie wykładniki są rozpatrywane
```

7. Zdefiniować funkcję `int strfind(String gdzie, String co)`, która szuka łańcucha `co` w łańcuchu `gdzie` i jeżeli go znajdzie, to jej wynikiem jest pozycja, na której ten łańcuch zaczyna się w łańcuchu `gdzie`. Jeżeli nie udało się znaleźć łańcucha to wtedy wynikiem ma być -1.

   Przykłady:

   ```
   strfind("Ala ma kota", "ma") - wynik to 4
   strfind("Ala ma kota", "Ala ma kota") - wynik to 0
   strfind("Ala ma kota", "") - wynik to 0, bo pusty łańcuch jest
                                podłańcuchem każdego innego łańcucha
   strfind("Pies", "jakis napis") - wynik to -1
   strfind("Ala ma kota", "pies") - wynik to -1
   ```

8. Napisać funkcję `int wordCount(String text)`, której wynikiem jest liczba wszystkich słów występujących w podanym jako parametr tekście. Do sprawdzania, czy dany znak tekstu jest „białym znakiem" można zastosować metodę `Character.isWhitespace`. Za słowo przyjmujemy każdy ciąg znaków niezawierający białego znaku.

9. Napisać funkcję `String[] podzielNaSlowa(String tekst)`, która dzieli podany tekst na słowa i zapisuje je w tablicy. Wynikiem funkcji jest tablica zawierające kolejno słowa z tekstu. Za słowo przyjmujemy każdy ciąg znaków niezawierający białego znaku.

10. Zdefiniować funkcję `int strFindAndCount(String gdzie, String co)`, która zlicza wystąpienia łańcucha `co` w łańcuchu `gdzie`. Jej wynikiem jest wyznaczona liczba wystąpień. Jeżeli nie udało się znaleźć łańcucha, to wtedy wynikiem jest, oczywiście, 0.

    Przykłady:

    ```
    strFindAndCount("Ala ma kota", "ma") - wynik to 1
    strFindAndCount("mama ma kota", "ma") - wynik to 3
    strFindAndCount("Ala mmaa ma kota", "ma") - wynik to 2
    strFindAndCount("Ala ma kota", "Asia") - wynik to 0
    ```

11. Zdefiniować procedurę `String strcut(String str, int start, int ile)`, która wycina z podanego łańcucha wszystko co znajduje się w podanym zakresie. Wynikiem funkcji powinien być łańcuch bez znaków znajdujących się na pozycjach od `start do start+ile-1`.

    Przykłady:

    ```
    strcut("Ala ma kota", 4, 3) - wynik to "Ala kota"
    strcut("Ala ma kota", 0, 4) - wynik to "ma kota"
    strcut("Ala ma kota", 0, 11) - wynik to ""
    ```

12. Napisać program, który wykorzystując część z zaimplementowanych wcześniej funkcji wyznacza:

    - Sumę wszystkich liczb znajdujących się w tablicy (jako liczbę traktuje się łańcuch, którego początkiem jest liczba - format jak w funkcji strToInt().
    - Łańcuch będący połączeniem wszystkich komórek tablicy, których wartość łańcucha nie jest liczbą (definicja liczby analogiczna do pkt. 1).
    - Liczbę wystąpień określonej frazy we wszystkich komórkach „nieliczbowych" tablicy.
    - Liczbę wystąpień określonej frazy w łańcuchu, o którym mowa w pkt. 2.
    - Stosunek wystąpień frazy w komórkach tablicy (pkt. 3) do liczby wystąpień w powstałym łańcuchu (pkt. 4).

    Przykład:

    ```
    Tablica, o której mowa w zadaniu:
    zadania[N][M]={"mamla", " mama ", "+12", "0001", "991-234-3",
                   "-12e5", "-12e-5", "+zonmakm", "ax2", "amakotma"};
                   // gdzie N=M=10;
    Szukana fraza:
    f[N]="ma";
    Wynik wyświetlony na konsolę:
    Pkt. 1: -1199008
    Pkt. 2: mamla mama +zonmakmax2amakotma
    Pkt. 3: 6
    Pkt. 4: 7
    Pkt. 5: 0.857143
    ```

13. Napisać funkcję `String poprzestawiaj(String tekst, int [] kolejnosc)`, której wynikiem jest łańcuch złożony ze znaków w zmiennej `tekst` ułożonych wg kolejności podanej w tablicy `kolejnosc`, tzn. i-ty znak tekstu powinien znaleźć się w wynikowym łańcuchu na pozycji `kolejnosc[i]`.

    Przykład, dla poniższego wywołania funkcji:

```
String tekst = "Egzamin";
int [] kol = { 0, 1, 4, 3, 2, 6, 5 };
System.out.println(poprzestawiaj(tekst, kol));
```

wynikiem powinien być łańcuch:

```
Egmazni
```

14. Napisać funkcję `boolean czyAnagram(String t1, String t2)`, która sprawdza, czy łańcuch `t2` jest anagramem tekstu `t1`, czyli czy składa się z tych samych znaków, ale ustawionych niekoniecznie w tej samej kolejności

    Uwaga, należy sprawdzać jedynie małe i duże litery alfabetu angielskiego, jednak bez względu na ich wielkość, tzn. zarówno małe 'a' jak i duże 'A' liczone są tak samo. Pozostałe znaki nie są sprawdzane, a więc nie mają wpływu na to, czy słowo będzie uznane za anagram innego.

    Przykładowo, dla poniższego fragmentu programu:

    ```
    System.out.println(czyAnagram("kolej", "olejk"));
    System.out.println(czyAnagram("kolej", "kole"));
    System.out.println(czyAnagram("kolej", "K O L E J"));
    System.out.println(czyAnagram("Gregory House", "Huge ego, sorry"));
    ```

    na ekranie powinno zostać wyświetlone:

    ```
    true
    false
    true
    true
    ```

15. W języku HTML oraz kaskadowych arkuszach stylów (CSS) powszechne jest ustalanie kolorów elementów w postaci łańcucha `#RRGGBB`, gdzie RR jest dwucyfrową liczbą (od 0x0 do 0xFF) w kodzie szesnastkowym oznaczającą ile czerwieni jest w wynikowym kolorze. Analogicznie GG oznacza nasycenie zieleni, a BB niebieskiego.

    Napisać funkcję `int [] HTMLColor2RGB(String color)`, która jako parametr przyjmuje łańcuch postaci "#RRGGBB" i zwraca tablicę 3 liczb całkowitych w systemie 10 oznaczających nasycenie poszczególnych składowych.

    Przykład

    ```
    Wynikiem HTMLColor2RGB("#FF0050") powinna być tablica { 255, 0, 80 }.
    Wynikiem HTMLColor2RGB("#001020") powinna być tablica { 0, 16, 32 }.
    ```

16. Dywan Sierpińskiego to fraktal otrzymany z kwadratu poprzez podzielenie go na dziewięć (3×3) mniejszych kwadratów i usunięcie środkowego kwadratu. Procedura ta stosowana jest rekurencyjnie do ośmiu powstałych w ten sposób mniejszych kwadratów.

    Napisać funkcję rekurencyjną, która wyznacza tekstowo–graficzną reprezentację dywanu Sierpińskiego po n krokach. Należy przyjąć, że rysowanie odbywa się w tablicy znaków o wymiarach $n^3 \times n^3$. Jeżeli dane pole ma być zamalowane wstawiany jest symbol #, w przeciwnym razie spacja. Powstałą w ten sposób tablicę znaków należy wypisać na ekranie.

    Przykład wykonania dla kolejnych wartości n.

    Dla $n = 1$:

```
#
```

    Dla $n = 2$:

```
#########
# ## ## #
#########
###   ###
# #   # #
###   ###
#########
# ## ## #
#########
```

    Dla $n = 3$:

```
###########################
# ## ## ## ## ## ## ## ## #
###########################
###   ######   ######   ###
# #   # ## #   # ## #   # #
###   ######   ######   ###
###########################
# ## ## ## ## ## ## ## ## #
###########################
#########         #########
# ## ## #         # ## ## #
#########         #########
###   ###         ###   ###
# #   # #         # #   # #
###   ###         ###   ###
#########         #########
# ## ## #         # ## ## #
#########         #########
###########################
# ## ## ## ## ## ## ## ## #
###########################
###   ######   ######   ###
# #   # ## #   # ## #   # #
###   ######   ######   ###
###########################
# ## ## ## ## ## ## ## ## #
###########################
```