1. Napisać program, który tworzy dwuwymiarową tablicę liczb całkowitych o losowej wielkości wymiaru (wylosowanej z zakresu [50, 100] i podzielnej przez 4), gdzie liczba wierszy jest równa liczbie kolumn. Następnie tablica zostaje wypełniona liczbami losowymi, z wyjątkiem elementów znajdujących się na przekątnych. Liczby mają być losowane z zakresu [a, b), gdzie liczby a i b podawane są przez użytkownika. Wartości na przekątnych mają być wypełnione w losowym układzie w 75% przypadków cyfrą 1, a w pozostałych 25% przypadków wartością -1.
Program powinien wyświetlić na ekran liczbę komórek, których wartość jest mniejsza od iloczynu indeksu wiesza i kolumny tej komórki.

---

2. Napisać funkcję, która w dwuwymiarowej tablicy łańcuchów znaków (podanej przez parametr) znajduje liczbę wystąpień frazy podanej jako drugi parametr. Jeśli w danej komórce fraza występuje kilkukrotnie, to należy policzyć każde jej wystąpienie.
Funkcja zwraca liczbę wystąpień frazy. Natomiast dodatkowo wyświetlona zostaje średnią długość łańcuchów znajdujących się w tabeli oraz łańcuch stworzony z konkatenacji trzech pierwszych* znaków łańcuchów leżących w kolumnach o indeksach podzielnych przez 5 i niepodzielnych przez liczbę przekazaną przez trzeci parametr funkcji.
* Jeśli łańcuch jest krótszy od 3 znaków, to należy zastosować wszystkie jego znaki.

---

3. Napisać funkcję split2, która działa podobnie, jak funkcja split (pozwalającej na podzielenie łańcucha znaków na pod łańcuchy (względem podanego znaku / ciągu znaków) i zwrócenie ich w postaci tablicy łańcuchów znaków), ale tym razem podział łańcucha dokonywany może być względem wielu znaków / ciągów znaków (przekazanych w tablicy znaków). Przykładowa deklaracja funkcji:
String[] split2(String str, String[] tStr);

---

**4. Pierwszy / ostatni.**

Zaimplementować funkcję `void firstlast(String s, char c)`, która oblicza liczbę znaków znajdujących się pomiędzy pierwszym i ostatnim wystąpieniem znak `c` w ciągu `s`. Funkcja musi wydrukować na standardowe wyjście obliczoną liczbę znaków oraz tekst znajdujący się pomiędzy wspomnianymi wystąpieniami znaku `c`. W przypadku, gdy szukany znak nie występuje w podanym ciągu lub występuje tylko raz, funkcja powinna przyjąć odległość równą -1.
Dla następujących wywołań funkcji:

```
firstlast(”a12345a6789a”, 'a');
firstlast(”cccbbccc”, 'b');
firstlast(”aaaaaa”, 'c');
```

na wyjściu powinien zostać wydrukowany tekst:

```
Odległość: 10. Tekst '12345a6789'
Odległość:  0. Tekst ''
Odległość: -1. Tekst ''
```

---

**5. Sortowanie**

Zaimplementować funkcję `void sortuj(String s)`, która drukuje na standardowym wyjściu posortowane (w kolejności rosnącej) wyrazy zapisane w ciągu znaków `s`. Wielkość liter nie ma znaczenia, w związku z czym np. `'a'=='A'`. Po wyświetleniu wyrazów należy także wyświetlić średnią długość wszystkich łańcuchów zapisanych w podanym ciągu znaków.
**Uwaga:** Dla utrudnienia wolno korzystać z funkcji standardowych toUpperCase() oraz to toLowerCase().

Dla następujących wywołań funkcji:

```
sortuj(”Aleksandra Joanna, Agnieszka”);
sortuj(”Ala ma kota i dwie agrafki”);
```

na wyjściu powinien zostać wydrukowany tekst:

```
'Agnieszka Aleksandra Joanna', 8,33
'agrafki Ala dwie i kota ma', 3,5
```

---

**6. Odwróć liczbę**

Zaimplementować funkcję z dwoma parametrami (liczby całkowita). W funkcji należy wyznaczyć moduł pierwszej liczby, a następnie odwróci kolejność cyfr, z których składa wyznaczony moduł. W funkcji powinna zostać zwrócona nowo powstała liczba, a dodatkowo poprzez drugi parametr suma pierwotnej i nowej liczby.
**Uwaga:** Nie wolno korzystać z funkcji dokonujących konwersji liczby na łańcuch znaków (szeroko rozumiany) i łańcuchów znaków na liczby.

Przykład:
```
5327
Zwrócone zostanie:
7235
12562
```

---

**7. Anagram**

Zaimplementować funkcję `boolean anagram(String a, String b)`, która sprawdza, czy wyrażenie `a` jest anagramem wyrażenia `b` (wyrażeniem utworzonym przez przestawienie liter alfabetu innego wyrażenia). Jeśli funkcja rozpozna anagram, to zwraca wartość prawdziwy (`true`), w przeciwnym wypadku zwraca wartość fałszywy (`fasle`).
**Uwaga:** Różnice w liczbie spacji występujących w ciągach nie mają znaczenia. Wielkość liter nie ma znaczenia, tzn. 'A'=='a'. Nie wolno używać funkcji standardowych zmieniających wielkości liter, należy więc samodzielnie ujednolicić ich wielkości w ciągach.

Dla następujących wywołań:
```
anagram(”warta”, ”trawa”);
anagram(” zwiedzam     patio   ”, ”tempo zadziwia”);
anagram(”abcdcba”, ”abcddcba”);
```
funkcja powinna zwrócić wartości:
```
true
true
false
```

---

**8. Akronim**

Zaimplementować funkcję `void akronim(String s)`, która drukuje na standardowym wyjściu akronim (wyraz sztucznie ułożony z pierwszych liter innych wyrazów). Należy pamiętać aby wyjściowy akronim był ułożony wyłącznie z wielkich liter.
**Uwaga:** nie wolno korzystać z funkcji standardowej toupper().
**Założenia:** Przekazywany łańcuch składa się tylko z poprawnych słów oddzielonych pojedynczą spacją (spacje nie występują w żadnych innych miejscach).

Dla następujących wywołań funkcji:
```
akronim(”Rzeczpospolita Polska”);
akronim(”Wydział Informatyki i Nauki o Materiałach”);
akronim(”Polski Związek Piłki Nożnej”);
```

na wyjściu powinien zostać wydrukowany tekst:
```
Akronim wyrażenia 'Rzeczpospolita Polska' to 'RP'
Akronim wyrażenia 'Wydział Informatyki i Nauki o Materiałach' to 'WIINOM'
Akronim wyrażenia 'Polski Związek Piłki Nożnej' to 'PZPN'
```