Zadanie 1 (10 pkt. na pracowni, później 5 pkt.). Napisz funkcję, która przyjmie jako argument tablicę liczb typu *double* (wraz z jej długością), znajdzie maksimum, minimum, i średnią arytmetyczną jej zawartości, i zapisze je w pozycjach, do których wskaźniki zostały podane jako trzy kolejne argumenty. Gdy tablica jest długości 0 (co w C jest zresztą, w ścisłym sensie, niemożliwe), działanie może być dowolne.

Napisz dwie funkcje, które przyjmą jako argument napis oraz znak i zwrócą wskaźnik na pozycję w zadanym napisie, która odpowiada najwcześniejszemu bądź najpóźniejszemu wystąpieniu zadanego znaku; jeśli znak nie występuje w napisie, należy zwrócić *NULL* (zdefiniowane w wielu nagłówkach, m.in. *stdlib.h*).

Napisz cztery funkcje, które przyjmą jako argument napis i zmodyfikują jego zawartość

- zamieniając wszystkie litery na wielkie,
- zamieniając wszystkie litery na małe,
- zamieniając go na tzw. *titlecase*, tj. wszystkie litery bezpośrednio po białym znaku (bądź na początku napisu) na wielkie, a pozostałe – na małe,
- odwracając go (najlepiej używając tylko stałej dodatkowej pamięci).

(Z tych samych powodów, co wzmiankowane na wykładzie, tych funkcji nie będzie dało się poprawnie wywołać na literałach napisowych – do testów będzie należało stworzyć napisy typu *char ...[]*. Wykorzystaj funkcje z *ctype.h*.)

W funkcji *main* umieść wywołania demonstrujące działanie powyższych funkcji. Czy te funkcje można jakoś składać? Co zrobić, żeby było to możliwe (na więcej sposobów)?

Zadanie 2 (10 pkt.). Napisz dwa programy – pierwszy z nich będzie generował wartości funkcji, a drugi drukował "wykres" funkcji o wartościach przeczytanych ze standardowego wejścia.

Program generujący ma wypisać na standardowe wyjście liczbę całkowitą *n*, a potem *n* liczb (niekoniecznie całkowitych, ale dla funkcji całkowitoliczbowej nie ma sensu wypisywać części ułamkowych) będących wartościami funkcji w kolejnych argumentach (możesz przyjąć, że są to argumenty od 0 do *n*-1). *n* ma być podawane jako argument wywołania, a jeśli go nie będzie, to domyślnie ma wynosić 80.

Funkcję wybierz sam(a) – może być dowolna, byle choć trochę ciekawa (liniowa o wybranych na stałe parametrach ciekawa nie jest). Jeśli zbiór jej wartości dla domyślnego zbioru argumentów to *Z*, to różnica pomiędzy maksimum i minimum zbioru $Z \cup \{0\}$ powinna być pomiędzy 10 a 40.

Jeśli chcesz, możesz coś tu dodatkowo skomplikować, np. użyć losowości (p. wyjaśnienia w zadaniu z poprzedniej listy) lub wczytywać dodatkowe parametry z parametrów wywołania czy (mniej ciekawie) standardowego wejścia.

Program drukujący ma przeczytać ze standardowego wejścia liczbę całkowitą *n*, a potem *n* liczb (w ogólności zmiennoprzecinkowych) i wydrukować wykres słupkowy o wartościach odpowiadających kolejnym liczbom (oczywiście po zaokrągleniu). Na wykresie ma być widoczna oś *X* zaznaczona znakami *'='*, a słupki (nad lub pod osią, w zależności od znaku wartości) należy zaznaczyć znakami *'#'* (trzeba też będzie oczywiście drukować odpowiednią liczbę spacji). Liczbę drukowanych wierszy dostosuj do ekstremów zbioru $Z \cup \{0\}$.

Jako argumenty wywołania program powinien akceptować:

- *-b* skutkujące wydrukowaniem dodatkowo obramowania wykresu, składającego się ze znaków *'|'* (segmenty pionowe obramowania), *'-'* (segmenty poziome obramowania), oraz *'+'* (narożniki obramowania, a także jego "przecięcia" z osią *X*); jeśli oś będzie na brzegu wykresu (tj. wartości po zaokrągleniu będą wszystkie nieujemne bądź niedodatnie) to należy potraktować ją jako część obramowania i nie drukować dodatkowego wiersza;
- *-m NAPIS* skutkujący użyciem kolejnych znaków z *NAPIS* zamiast wyżej wymienionych jako elementu słupka, osi *X*, i ew. pionowego, poziomego, i narożnego obramowania; *NAPIS* powinien mieć długość 2, a jeśli podano *-b*, to 5.

Argumenty są opcjonalne i mogą występować w dowolnej kolejności, a ich niepoprawna postać (np. brak *NAPIS* po *-m* lub jego zła długość, kilkukrotne powtórzenie *-m*, rzeczy inne niż opisane) powinna skutkować natychmiastowym wyjściem z programu.

Drukowanie wykresu możesz zaprogramować bez użycia dodatkowej pamięci na jego przygotowanie, ale jeśli wolisz jej użyć, to zadbaj o to, by samo drukowanie było tak proste jak instrukcja *for (int i = 0, i < m, i++) puts(wykres[i]);* gdzie (jak widać) *wykres* to tablica napisów. Dałoby się zresztą zrobić to tak, by wystarczył pojedynczy *puts* (i to bez rezygnowania z wygody, jaką daje dwuwymiarowa tablica), ale to może trochę przesada.

Program generujący funkcję nie powinien wypisywać nic, czego nie umie przetworzyć program drukujący wykres, tj. powinno się je dać wywołać dosłownie przekierowując potok wyjściowy pierwszego na wejściowy drugiego, bez żadnej "ręcznej" interwencji. Pliki źródłowe powinny nazywać się odpowiednio *gen.c* oraz *graph.c*.

Zadanie 3. Dane są liczby *n* i *m* oraz posadzka składająca się z kwadratowych kafelków w *m* kolumnach i *n* wierszach. Na każdym kafelku stoi liczba naturalna oznaczająca karę za pobrudzenie ubłoconym obuwiem świeżo mytego kafelka.

Zaczynamy na lewo od pierwszej kolumny kafelków i przechodzimy przez posadzkę, robiąc kroki zawsze o 1 kolumnę w prawo, albo pozostając w tym samym wierszu, albo zmieniając go na sąsiedni (powyżej lub poniżej); w pierwszym kroku możemy wejść na dowolny kafelek w pierwszej kolumnie. Chcemy przejść przez posadzkę na jej prawą stronę, płacąc możliwie małą karę.

Wejście

W pierwszej linii wejścia znajdują się liczby *n* i *m*, *n* nie większe od 1000.

W każdej z kolejnych *n* linii wejścia znajduje się *m* liczb naturalnych, nie większych od 1000. Stanowią one wartości kar odpowiadających kolejnym kafelkom w danym wierszu.

Wyjście

Na wyjściu powinna się pojawić jedna liczba naturalna oznaczająca minimalną karę, jaką należy zapłacić.

Przykłady

A

Wejście:

```
2 3
1 2 3
2 1 5
```

Wyjście

```
5
```

(Optymalne jest przejść przez kafelki o karach 1, 1, 3)

B

Wejście:

```
2 3
1 7 9
2 8 4
```

Wyjście

```
12
```

C

Wejście:

```
3 4
1 1 1 1
2 3 1 7
1 1 2 8
```

Wyjście

```
4
```