Zadanie 1. Napisz program, który przeczyta pliki, których nazwy zostaną podane jako argumenty wywołania, odczyta zawarte w nich informacje geograficzne o odcinkach szlaków (turystycznych, krajoznawczych), i wypisze na standardowe wyjście kilka informacji wyliczonych na ich podstawie.

Pliki, których przykłady załączone są do zadania, zapisane są w formacie GPX (czyli pewnym schemacie XMLa), w dodatku są dosyć ładnie sformatowane i zawierają tylko znaki ASCII. XML to język znaczników pozwalający na gromadzenie tekstowych danych w pewnej drzewiastej strukturze. W naszym przypadku struktura (a raczej ta jej część, która nas w tym zadaniu interesuje), wygląda tak (porównaj ten schemat z zawartością któregoś z plików):

```
gpx
 metadata
 wpt
 ...
 wpt
 trk
  name
  link
  trkseg
   trkpt
   ...
   trkpt
```

Czyli: w każdym pliku znajduje się jeden element gpx, w którym zagnieżdżone są elementy metadata (który sam ma wewnętrzną strukturę, ale ta część nas nie interesuje), wpt (skrót od waypoint), oraz trk, który z kolei zawiera elementy name, link i trkseg, a ten ostatni wreszcie zawiera pewną liczbę elementów trkpt (tu także zawsze co najmniej 2).

Zarówno elementy wpt jak i trkpt mają atrybuty lat oraz lon (skróty od odpowiednio latitude oraz longitude), i zawierają m.in. element ele (skrót od elevation; zawiera napis) – te trzy informacje opisują przestrzenne położenie danego punktu. Na tej podstawie będziemy liczyć długości szlaków, traktując je jako linie łamane (otwarte), których kolejne wierzchołki to elementy trkpt zapisane w danym pliku.

Program ma wypisywać maksimum, minimum, oraz medianę długości szlaków, a także nazwy plików oraz nazwy szlaków (odczytane z elementu name w elemencie trk), które te wartości osiągają (w przypadku remisu – zresztą mało prawdopodobnego dla obliczeń na liczbach zmiennoprzecinkowych – można wybrać dowolnego z "kandydatów"). Uwaga: dla populacji parzystej mocy czasem przyjmuje się, że mediana to średnia arytmetyczna dwóch "środkowych" wartości – tu bardziej naturalne będzie przyjąć, że w takich sytuacjach bywają dwie mediany (i znowu wybrać dowolną z nich).

Dodatkowo, dla każdego szlaku należy określić stosunek jego długości do długości pojedynczego odcinka łączącego skrajne punkty i wypisać największy skończony spośród takich stosunków, znowu wraz z nazwą pliku i nazwą szlaku.

Wersja uproszczona programu (do 8 pkt.) może korzystać z ładnego formatowania i przede wszystkim poprawności przykładowych plików i np. "w ciemno" czytać miejsca, w których pojawia się napis <trkpt itp. (natomiast nawet wtedy akceptuj niecałkowite wartości wysokości). Więcej będzie można dostać za bardziej "ostrożne" podejścia, a na komplet punktów należy – wciąż w pewien bardzo ograniczony sposób – sprawdzać strukturalną poprawność całego dokumentu XML. Wtedy należałoby:

przy wejściu do znacznika (znak <) oczekiwać, że ten znacznik się kiedyś skończy (znak >);
przy wejściu do wartości atrybutu (znak " po nazwie atrybutu i znaku równości) oczekiwać, że ona też się kiedyś skończy, ale uważać na znaki ucieczki (\");
przy wejściu do elementu oczekiwać, że ten element się kiedyś skończy (odpowiednim znacznikiem zamykającym), ale mogą być w nim zagnieżdżone inne elementy (może się przydać stos nazw pozagnieżdżanych elementów, w których w danym momencie jesteśmy) – i uważać na elementy nieposiadające zawartości (w naszych plikach jest ich tylko jeden rodzaj – bounds wewnątrz metadata)
liczyć się z odstępstwami od ww. schematu (dwa elementy trk, brak elementów trkpt – bo obecność jednego trkpt da się jeszcze w miarę sensownie zinterpretować, brak któregoś z atrybutów lat i lon, których oczekiwaliśmy)...
Podział kodu na funkcje i moduły jest do Twojej decyzji (która też wpływa na punktację). Jedynym wyjątkiem jest funkcja do liczenia długości odcinka na podstawie współrzędnych geograficznych i wysokości (oczywiście w metrach), która musi być w osobnym module – tu zaimplementujemy wersję bardzo prymitywną i proszącą się o pilną poprawkę. Skorzystaj z twierdzenia Pitagorasa w trzech wymiarach i załóż, że sekunda kątowa szerokości geograficznej odpowiada 30,72 m, a sekunda długości geograficznej (na szerokości geograficznej... Greenwich, ale to akurat leży gdzieś pomiędzy wartościami z przykładowych plików) – 19,22 m. Wartości w plikach są podane w stopniach, a stopień to 3600 sekund.

Oczekiwane wywołanie programu to ./a.out *.gpx po umieszczeniu go w jednym katalogu z przykładowymi plikami (oczywiście ściągnij wersję z końcami wierszy odpowiednimi dla Twojego systemu operacyjnego). Jeśli okaże się, że Twój system operacyjny nie pozwala uruchomić programu z kilkunastoma tysiącami argumentów wywołania, ogranicz się do... możliwie wielu. Uwaga: system operacyjny prawie na pewno nie pozwoli Ci otworzyć jednocześnie kilkunastu tysięcy plików. Zauważ, że po przetworzeniu danego pliku wystarczy zapamiętać na jego temat kilka liczb i jakiś napis.

Zadanie 2. Chcielibyśmy zakodować pewien tekst przy pomocy kodowania Huffmana (https://pl.wikipedia.org/wiki/Kodowanie_Huffmana). Drzewo będziemy generować z

podanych na wejściu 26 liczb odpowiadających spodziewanej częstotliwości kolejnych liter alfabetu. Tekst dany jest w postaci ciągu małych liter alfabetu angielskiego.

Przy generowaniu drzewa należy korzystać z następujących zasad:
- na początku sortujemy litery malejąco po spodziewanych częstotliwościach (jeśli dwie litery mają te same częstotliwości, ich kolejność powinna być alfabetyczna) i tworzymy jednoelementowe drzewa.
- nowe drzewo zawsze generujemy z dwóch ostatnich na liście, ostatni element zostaje prawym poddrzewem nowego drzewa, a przedostatni lewym. To drzewo jest teraz ostatnie na liście
- przesuwamy ostatnie drzewo na liście w lewo tak długo, aż lista drzew nie będzie posortowana po częstotliwościach (jeśli na liście pojawią się dwie równe częstotliwości w węzłach drzewa, nie należy ich zamieniać).
Dodatkowo w kodowaniu lewe poddrzewo odpowiada znakowi 0 a prawe 1.

Wejście:
W pierwszej linii wejścia znajduje się 26 liczb naturalnych, co najmniej 2 dodatnie (możesz założyć, że ich suma mieści się w typie danych int) odpowiadających spodziewanym częstotliwościom wystąpienia kolejnych liter alfabetu angielskiego. W drugiej linii wejścia znajduje się liczba naturalna n ≤ 106. Trzecia linia wejścia składa się z n małych liter alfabetu angielskiego (możesz założyć, że tylko tych liter, które miały niezerowe częstotliwości).

Wyjście:
Na wyjściu powinien znaleźć się ciąg zer i jedynek odpowiadający zakodowanej wiadomości.

Przykłady:
A
Wejście:

12 7 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0

6
abnnab

Wyjście:

1000101100

B
Wejście:

1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4

abcd

Wyjście:

10110001

C
Wejście:

12 2 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0
6
abnnab

Wyjście:

0111010011