Zadanie 1

Zdefiniuj w programie dowolny napis, czyli tablicę znaków, na przykład za pomocą instrukcji

```
char tab[] = "To be, or not to be, that is the question"
```

Program powinien:

1. sprawdzić, czy wszystkie znaki są literami, odstępami, lub znakami kropki lub przecinka; jeśli nie, to wykonanie powinno zostać przerwane;
2. wypisać ilość *liter* w napisie (nie licząc odstępów, kropek i przecinków);
3. wypisać ilość liter *różnych* w napisie. Wielkość liter nie powinna być brana pod uwagę, czyli na przykład 'b' i 'B' to ta sama litera. Tak więc w napisie `"Warszawa"` jest 8 liter, ale tylko 5 różnych;
4. wypisać literę, która w napisie pojawia się największa liczbę razy oraz jej liczbę wystąpień; bez rozróżniania liter dużych i małych;
5. wypisać napis, opuszczając znaki odstępu, kropki, przecinka oraz litery które pojawiły się w tym napisie wcześniej (znowu, bez rozróżniania liter dużych i małych). Na przykład, dla napisu `"Zaraz wracam"` powinno zostać wypisane `"ZARWCM"`.

UWAGA:
Nie wolno włączać do programu nagłówków ***cstring*** *(****string.h****) i/lub* ***string****.*

Rozmiar napisu może być dowolny (nie wolno zakładać, że jest mniejszy od jakiejś maksymalnej liczby).

Wolno definiować pomocnicze tablice, ale tylko statyczne (o rozmiarze znanym na etapie kompilacji – czyli „wbitym" do kodu).

Można założyć, że kody odpowiadające literom są kodami ASCII, to znaczy wartości liczbowe odpowiadające 'A', 'B',...,'Z' są kolejnymi liczbami całkowitymi, i tak samo 'a', 'b',..., 'z'. Wartości zmiennych znakowych są w wyrażeniach konwertowane do wartości całkowitych, na przykład `'a'-7` *wynosi 90, bo kod ASCII litery 'a' to 97. Zauważmy, że 'a'-'A' wynosi 32 i taka też jest dla wszystkich liter różnica między kodem odpowiadającym odpowiedniej literze małej i dużej.*

Poszczególne zadania powinny być realizowane przez odpowiednie funkcje; można też definiować dodatkowo przydatne funkcje pomocnicze.

Nie stosuj liter polskich — wyłącznie standardowe litery alfabetu łacińskiego.

Na przykład program

```
#include <iostream>
using namespace std;

// funkcje pomocnicze...
```

```
bool isLegal(const char* s) {
    // ...
}

int numbOfLetters(const char* s) {
    // ...
}

int numbOfDiffLetters(const char* s) {
    // ...
}

void mostFrequentLetter(const char* s) {
    // ...
}

void allLetters(const char* s) {
    // ...
}

int main() {
    const char str[] =
                "To be, or not to be, that is the question";
    // 1.
    bool legal = isLegal(str);
    if (!legal) {
        cout << "Illegal string" << endl;
        return 1;
    }
    cout << "String OK" << endl;

    // 2.
    int letters = numbOfLetters(str);
    cout << letters << " letters" << endl;

    // 3.
    int diffLett = numbOfDiffLetters(str);
    cout << diffLett << " different letters" << endl;

    // 4.
    mostFrequentLetter(str);

    // 5.
    allLetters(str);
```

```
}
```

powinien wypisać

```
String OK
30 letters
12 different letters
T/t occurs 7 times
TOBERNHAISQU
```

Zadanie 2

Przeprowadzamy ankietę na pewien temat. Wynik pojedynczej ankiety (uzyskanej od jednego respondenta) zawiera następujące informacje, które należy zakodować w *jednej* zmiennej typu **unsigned short** (można założyć, że ma ona 2 bajty, czyli 16 bitów):

1. płeć — 1 bit, bo 2 możliwości (kobieta, mężczyzna), kodowane jako 0 lub 1;
2. stan cywilny — 2 bity, bo 4 możliwości (panna/kawaler, mężatka/żonaty, rozwódka/rozwodnik, wdowa/wdowiec) kodowane jako 0, 1, 2 lub 3;
3. grupa wiekowa — 2 bity, bo 4 możliwości (np. 18-30, 31-45, 46-60, 60+) kodowane jako 0, 1, 2 lub 3;
4. wykształcenie — 2 bity, bo 4 możliwości (np. podstawowe, średnie, licencjat, magister+) kodowane jako 0, 1, 2 lub 3;
5. miejsce zamieszkania — 2 bity, bo 4 możliwości (np. wieś, miasto do 50 tys., miasto 50-400 tys., miasto ponad 400 tysięcy mieszkańców) kodowane jako 0, 1, 2 lub 3;
6. region kraju — 4 bity, bo (przypuśćmy) jest 16 regionów ponumerowanych od 0 do 15;
7. odpowiedź na pytanie ankietera – 3 bity, bo w ankiecie (przypuśćmy) było 8 możliwych odpowiedzi, ponumerowanych od 0 do 7.

Napisz funkcję

```
unsigned short koduj(int plec, int stan_cyw,
                     int grupa_wiek, int edu,
                     int zam, int region,
                     int odp);
```

która pobiera 7 liczb (jak wyżej), koduje informacje w nich zawarte w *jednej* zmiennej typu **unsigned short** i zwraca tę jedną liczbę do funkcji wywołującej.
Napisz również funkcję

```
void info(unsigned short kod);
```

która pobiera jeden argument typu **unsigned short** zawierający informacje o jednej ankiecie i wypisuje te informacje, np. w formie:

```
plec:           0
stan cywilny:   3
grupa wiekowa:  2
wyksztalcenie:  3
miejsce zam.:   0
region:        12
odpowiedz:      6
```

Nie używaj żadnych narzędzi z biblioteki standardowej (innych niż te z *iostream* do wypisywania wyników). Zamiast **unsigned short** możesz użyć **uint16_t** (chociaż na większości platform są to synonimy).

Zadanie 3

Napisz funkcję

```
bool checkpass(const char* pass);
```

która pobiera hasło (jako C-napis, czyli tablicę znaków) i sprawdza jego poprawność. Przyjmujemy, że poprawne hasło zawiera

1. co najmniej 8 znaków;
2. co najmniej 6 znaków różnych;
3. co najmniej 1 cyfrę;
4. co najmniej 1 dużą literę;
5. co najmniej 1 małą literę;
6. co najmniej 1 znak niealfanumeryczny (nie będący literą ani cyfrą).

Funkcja zwraca **true** jeśli hasło jest poprawne, a jeśli nie, to zwraca **false**, ale przedtem wypisuje komunikat o *wszystkich* przyczynach niepoprawności. Można założyć, że znaki są znakami ASCII o kodach w zakresie $[32, 126]$. [Może być przydatne zdefiniowanie też prostych funkcji pomocniczych.]

Na przykład program następujący

download *CStringPass.cpp*

```
#include <iostream>

// ...

bool checkpass(const char* pass) {
    // ...
}

int main() {
    using std::cout; using std::endl;
    const char* passes[] =
        {
            "AbcDe93",  "A1b:A1b>",  "Ab:Acb<",
            "abc123><", "Zorro@123", nullptr
```

```
        };
    for (int i = 0; passes[i] != nullptr; ++i) {
        cout << "checking " << passes[i] << endl;
        if (checkpass(passes[i])) cout << "OK" << endl;
        cout << endl;
    }
}
```

powinien wypisać coś w rodzaju

```
checking AbcDe93
Too short
No non-alphanumeric character

checking A1b:A1b>
Too few different characters

checking Ab:Acb<
Too short
Too few different characters
No digit

checking abc123><
No uppercase letter

checking Zorro@123
OK
```

Nie włączaj żadnych plików nagłówkowych innych niż *iostream*!
