Wstęp do programowania w języku C

Grupa TDr

Lista 4.

1. *(10 punktów w trakcie pracowni, później 5 punktów)*

Napisz funkcję o sygnaturze `void rewrite_string_as_utf8(char text[]);`, która w tekście `text` podmieni specjalne sekwencje znaków (pierwszy wiersz tabelki), na odpowiednie znaki (drugi wiersz tabelki) w kodowaniu UTF-8 (kody unikodowe podane są trzecim wierszu tabelki). Więcej o kodowaniu UTF-8 można przeczytać np. tutaj.

`A	`a	`C	`c	`E	`e	`L	`l	`N	`n	`O	`o	`S	`s	`X	`x	`Z	`z
Ą	ą	Ć	ć	Ę	ę	Ł	ł	Ń	ń	Ó	ó	Ś	ś	Ź	ź	Ż	ż
260	261	262	263	280	281	321	322	323	324	211	243	346	347	377	378	379	380

Przykładowo wywołanie poniższej funkcji `main` powinno wypisać dwukrotnie to samo.

```
int main() {
    char text1[] = "za`z`o`l`c g`e`sl`a ja`x`n ZA`Z`O`L`C G`E`SL`A JA`X`N";
    char text2[] = u8"zażółć gęślą jaźń ZAŻÓŁĆ GĘŚLĄ JAŹŃ";
    rewrite_string_as_utf8(text1);
    printf("%s\n%s\n", text1, text2);
}
```

Możesz wykorzystać funkcję `write_utf8_char` (podaną na końcu listy zadań), która do tablicy `dst` wpisuje bajty kodujące znak o kodzie `n` i zwraca liczbę wpisanych bajtów.

Aby uniknąć pisania kilkunastu instrukcji warunkowych, możesz stworzyć tablice odpowiednich znaków i kodów i przeglądać je za pomocą pomocniczej pętli.

Śmiało zadawaj pytania prowadzącemu.

2. *(10 punktów, zadanie sprawdza zdalnie Mateusz Wasylkiewicz)*

Napisz program, który wczyta liczbę naturalną n ($0 \leqslant n \leqslant 255$) i wypisze regułę parkietowania kodowaną przez nią oraz poniżej trzy kody równoważnych reguł (opis kodowania i równoważności znajduje się poniżej przykładów).

Przykładowo dla liczby 124 i kodowania 1 jako kropki `'.'`, a 0 jako kratki `'#'`, program powinien wypisać regułę z poprzedniej listy zadań oraz liczby:

```
...  ..#  .#.  .##  #..  #.#  ##.  ###
 #    .    .    .    .    .    #    #
```

```
110 193 137
```

a dla liczby 30:

```
...  ..#  .#.  .##  #..  #.#  ##.  ###
 #    #    #    .    .    .    .    #
```

```
86 135 149
```

Jeśli 1 zakodujemy jako `'1'`, a zero jako `'0'`, to dla liczby 110 powinien wypisać:

```
111  110  101  100  011  010  001  000
 0    1    1    0    1    1    1    0
```

```
124 137 193
```

Opis kodowania reguły. Reguła parkietowania ma być prezentowana w postaci takiej, jak powyżej, czyli: Nagłówek składa się z ośmiu trzyznakowych zestawów kodujących binarnie liczby i od 7 do 0. Ma być wyświetlany zawsze w tej samej kolejności (malejącej) z odstępami. Poniżej, w odpowiednich miejscach, mają być bity i-tych potęg dwójki liczby n.

Opisy równoważności. Pierwszy równoważny kod ma kodować regułę parkietującą lustrzane odbicie w pionie (lewa zamienia się z prawą) reguły kodowanej przez n.

Drugi równoważny kod ma kodować regułę parkietującą negatyw (zamiana jedynek z zerami) reguły kodowanej przez n (jeśli zaczyna od negatywu konfiguracji początkowej).

Trzeci równoważny kod ma kodować regułę przekształconą i przez odbicie lustrzane, i przez negatyw.

Wyniki możesz testować porównując się z WolframAlpha, do którego odsyłają łącza umieszczone w treści zadania.

Zauważ, że kropka i kratka nie są podawane na wejściu programu — są one ustalone „na sztywno” w kodzie. Zorganizuj kod tak, aby dało się je zmieniać zmieniając po jednym znaku w kodzie (bez tego 2 punkty mniej).

Do operacji na bitach używaj operacji bitowych (a nie np. mnożenia i dzielenia, bez tego 3 punkty mniej).

3. *(10 punktów)*

w serwisie SKOS

```
int last(int n) {
    return ((1<<n)-1);
}

int write_utf8_char(char dst[], int n) {
    if (n < (1<<7)) {
        dst[0] = n;
        return 1;
    }
    if (n < (1<<11)) {
        dst[0] = (last(2)<<6) | ( n>> 6);
        dst[1] = (      1<<7) | ( n     &last(6));
        return 2;
    }
    if (n < (1<<16)) {
        dst[0] = (last(3)<<5) | ( n>>12);
        dst[1] = (      1<<7) | ((n>> 6)&last(6));
        dst[2] = (      1<<7) | ( n     &last(6));
        return 3;
    }
    if (n < (1<<21)) {
        dst[0] = (last(4)<<4) | ( n>>18);
        dst[1] = (      1<<7) | ((n>>12)&last(6));
        dst[2] = (      1<<7) | ((n>> 6)&last(6));
        dst[3] = (      1<<7) | ( n     &last(6));
        return 4;
    }
    return 0;
}
```