Komunikat

We wszystkich zadaniach należy używać zapisu wskaźnikowego. Zapis tablicowy (użycie znaków [ ]) możliwy jest tylko przy inicjalizacji tablicy.

W pierwszej linijce w funkcji main() wstaw linijkę:

setvbuf(stdout, NULL, _IONBF, 0);.

Spowoduje ona, że nie będzie problemów z opóźnionym wyświetlaniem komunikatów wypisywanych przez program.

Zadanie 1 (2pkt)

Mandalorian Code: This is the way.

W odległej galaktyce, w czasach, gdy Mando patrolował przestrzeń kosmiczną, napotkałeś tajemniczy artefakt. Aby go zidentyfikować, postanowiłeś użyć technologii znanej jako "Mandalorian Code".

Wykorzystując potęgę wskaźników napisz program w który będzie odkrywał kod ukryty w znakach podczas misji poprzez generowanie losowego ciąg znaków w zakresie „A-Z" o długości podanej przez użytkownika z zakresu od 1 do „MAX_LENGTH", jeżeli zostanie podana liczba z poza zakresu program ma zakończyć działanie i zwrócić wartość „1". Wygenerowany ciąg znaków powinien zostać wyświetlony.

```
void generateRandomString(char *str, int length);
 Podaj dlugosc kodu znakow z Mando Misji (max 100):15
  Wygenerowany ciag znakow: JETKKAIUMNGUKPK
```

Następnie, korzystając z Mando-skryptu w celu sprawdzenia mocy znaków, przeprowadzisz analizę, aby odnaleźć takie same znaki i poznać ich tajemniczą moc. Program ma przeprowadzić analizę ciągu znaków i odnaleźć takie same znaki w celu ich zliczenie, czyli podania sumy wystąpień każdego znaku. Podsumowanie ma zostać wyświetlone w formie zestawienia wszystkich pozycji.

```
void findAndDisplayStats(char *str);
 Statystyki:
 Znak A moc 1 MandoPower
 Znak E moc 1 MandoPower
 Znak G moc 1 MandoPower
 Znak I moc 1 MandoPower
 Znak J moc 1 MandoPower
 Znak K moc 4 MandoPower
 Znak M moc 1 MandoPower
 Znak N moc 1 MandoPower
 Znak P moc 1 MandoPower
 Znak T moc 1 MandoPower
 Znak U moc 2 MandoPower
```

Kolejnym zadaniem programu będzie usunięcie z ciągu wszystkich znaków, których ilość wystąpień jest liczbą parzystą. To jakbyś odfiltrował słabe elementy, aby ujawnić prawdziwą moc kodu. Na koniec, uzbrojony w nową wiedzę, program wyświetli zmodyfikowany ciąg znaków. Teraz masz narzędzie do odczytania ukrytych wiadomości w galaktyce.

```
Tajemny kod mocy to: JETAIMNGP
```

Teraz masz narzędzie do odczytania ukrytych wiadomości w galaktyce.

Zadanie 2 (3pkt)

W odległej galaktyce, na planecie, gdzie kod ma moc, zlecono Ci zadanie stworzenia programu, który wykorzystuje magiczną rekurencję i potężne wskaźniki, aby wygenerować tablicę 10x10 literami małymi oraz dużymi. Twój celem jest posortowanie każdego z wierszy od 'z' do 'A' - jak prawdziwy mistrz kodu w galaktyce.

Twoja misja składa się z następujących kroków:

1. Program ma wygenerować tablicę 10x10 literami małymi oraz dużymi za pomocą rekurencji i wskaźników. To jak odkrywanie tajemniczych symboli na starożytnych kamieniach.

```
void fillArray(char (*arr)[COLS], int rows, int cols);
```

2. Następnie, korzystając z mocy kodu, program ma wyświetlić tablicę przed sortowaniem. To jak ujrzenie nieznanego alfabetu na tajemniczej ścianie.

```
Nieznany alfabet na tajemniczej scianie:
   Q    H    W    e    g    j    O    Q    X    U
   N    M    b    b    Y    B    j    R    W    U
   J    b    p    l    E    Y    T    U    i    s
   D    Z    J    v    E    e    M    i    M    m
   y    n    R    d    v    P    E    K    J    Z
   Z    n    f    O    Y    c    G    F    Z    Y
   V    a    n    C    s    V    H    z    X    N
   c    v    n    h    a    j    E    F    L    V
   x    w    z    r    q    j    e    I    O    g
   h    n    c    C    E    w    K    S    n    P
```

3. Po tym etapie, program ma przystąpić do sortowania każdego wiersza od 'z' do 'A'. To jak ułożenie liter w magicznym szyfrze.

```
void recursiveSortArray(char (*arr)[COLS], int row, int start, int end);
```

4. Ostatecznie, uzbrojony w magiczne narzędzia, program ma ponownie wyświetlić tablicę po sortowaniu. To jak ujawnienie ukrytych znaczeń w starożytnym tekście.

```
Ukryte zanczenia w starozytnym tekscie:
   j    g    e    X    W    U    Q    Q    O    H
   j    b    b    Y    W    U    R    N    M    B
   s    p    l    i    b    Y    U    T    J    E
   v    m    i    e    Z    M    M    J    E    D
   y    v    n    d    Z    R    P    K    J    E
   n    f    c    Z    Z    Y    Y    O    G    F
   z    s    n    a    X    V    V    N    H    C
   v    n    j    h    c    a    V    L    F    E
   z    x    w    r    q    j    g    e    O    I
   w    n    n    h    c    S    P    K    E    C
```

Czy jesteś gotowy na tę niezwykłą misję?

Niech moc kodu będzie z Tobą.

Zadanie 3 (4pkt)

Napisz program rekurencyjnie, który wczyta obraz buty.bmp o wielkości 1000x1000 pikseli, a następnie podzieli go na nienakładające się okna o wielkości 250x250 pikseli. W danym oknie sprawdź czy ilość pikseli, których wartość jest większa bądź równa 200 , przekracza 50% wszystkich pikseli w danym oknie, to program narysuje ramkę wewnętrzną wokół tego okna o grubości 1 piksela i wartości 255. Jeśli ilość wystąpień wartości nie przekroczy 50% nie dokonuje się zmian na oknie. Każdy piksel ma wartości z przedziału 0-255, gdzie 0 to czarny, a 255 biały.

Skorzystaj z plików zamieszczonych w zadaniu:

-buty.bmp,

-bmp_io.h,

-bmp_io.c,

-main.c.

Przykład działania programu:

Zmieniono okno (0, 0) - (249, 249).

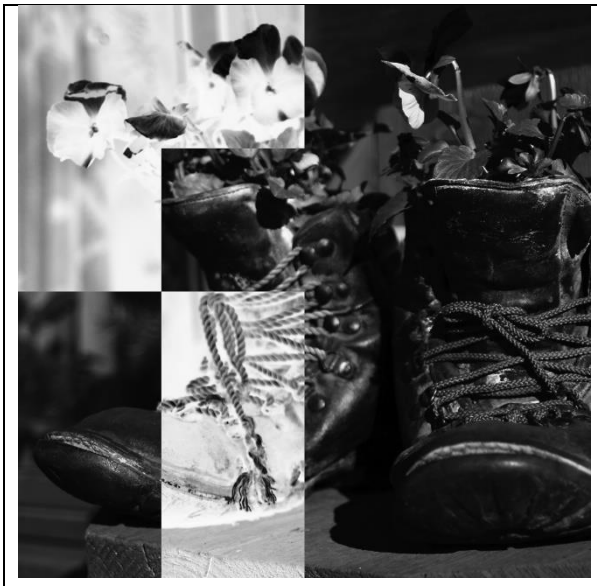
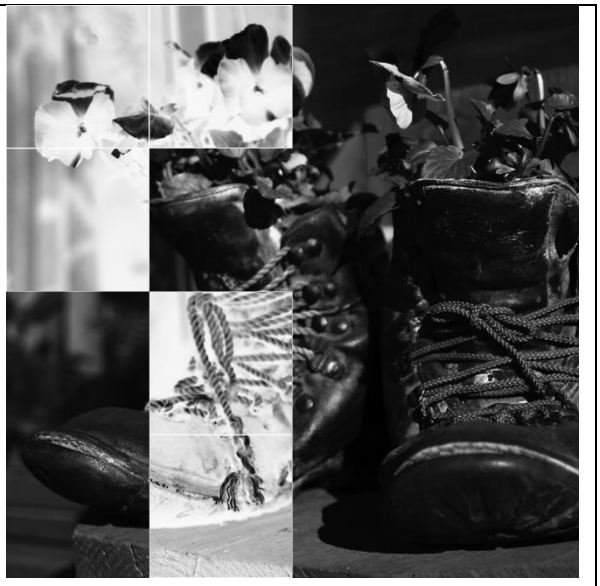
Zmieniono okno (0, 250) - (249, 499).

Zmieniono okno (250, 0) - (499, 249).

Zmieniono okno (250, 500) - (499, 749).

Zmieniono okno (250, 750) - (499, 999).

Przykład działania programu:

	
Obraz wejściowy	Obraz wynikowy

Przeliczenie punktów na ocenę:

- 0 - 4.5 punktu → ocena 2
- 5 - 5.5 punktu → ocena 3
- 6 - 6.5 punktu → ocena 3,5
- 7 - 7.5 punktu → ocena 4
- 8 - 8.5 punktu → ocena 4,5
- 9 punktów – ocena 5