tБележки:

- Желателно е всички функции да извеждат на екрана подробна информация за случващото се. Това ще помогне за проследяване на проблеми и ще направи резултата от изпълнението по-интересен.
- След написването на всяка функция или структура ги изтествайте
- Внимавайте за индексите в масивите
- Някои от функциите приемат като аргумент структура, която о в C е **копие** на оригиналната променлива. За да можете да направите промени по него и да ги видите извън функцията трябва да върнете променената променлива.
  Пример:

```
struct test_t do_stuff(struct test_t test_data) {
      test_data.value = 10;
      return test_data;
}
```

- не се плашете от задачите. Дори и да не можете да направите цялата задача направете колкото можете от нея. Всяка изпълнена и работеща подточка ще носи някакви точки

Задача 1.

1. Да се напише структура за оръжие, която има членове за модел, точност и размер на пълнителя
   a. `char model[128]`
   b. `int accuracy` - число от 0 до 100 означаващо процент на точни изстрели
   c. `int ammo_capacity`
2. Оръжията ще се изпробват срещу кръгла мишена с 10 разделения, които носят точки съответно 10 в центъра и 0 по ръба на мишената. Стрелящият винаги се цели точно в центъра, но може да се отклони в зависимост от точността на оръжието. Да се напише функция, която получава за аргументи точността на оръжието и точността на конкретен изстрел и връща като резултат получения резултат.
   Ако `shot_accuracy` е в интервала от 0 до `gun_accuracy`, то той е уцелил в центъра на мишената и резултатът е 10. Ако `shot_accuracy > gun_accuracy` то той е извън центъра и носи точки спрямо това колко далече е попаднал.

```
int get_score(int gun_accuracy, int shot_accuracy);
```

   Пример:

При точност т 40% `shot_accuracy` от 0 до 40 винаги носи 10 точки. От 41 до 46 ще носи 9, от 47 до 52 ще носи 8, от 53 до 58 ще носи 7 и т.н.

2*. В горния пример зависимостта между точността извън центъра и резултата е линейна. Променето пресмятането така, че зависимостта да е по-близка до гаусово разпределение (графиката е параболична, а не права).

3. Да се напише функция за тестване на оръжие, която получава за аргументи оръжие и брой изстрели. За всеки изстрел да се вземе случайна стойност от 0 до 100, която ще представлява точност на изстрела. Като се използва `get_score` да се изпише на екрана точността и резултата.

```
void test_gun(struct gun_t gun, int shot_count);
```

4. Да се модифицира `test_gun` така че след всички изстрели да извежда и общия резултат и средностатистическия.

Задача 2.

1. Да се напише структура за дърво, която има членове за височина и възраст
   a. `float height`
   b. `int age`
2. Да се напише структура за гора, която има членове за брой дървета и масив от дървета
   a. `int tree_count`
   b. `struct tree_t trees[10]`
3. Да се напише функция за увеличаване на възрастта на дърветата в дадена гора и пресмятане на новата им височина. Функцията увеличава age с 1, а пък височината със случайно число (с плаваща запетая) между 0.1 и 2.0.

```
struct forest_t age_trees(struct forest_t forest);
```

4. Да се напишат функции за извеждане на екрана на единично дърво и на цялата гора. Функцията за извеждане на цялата гора да използва тази за извеждане на единично дърво.

```
void print_tree(struct tree_t tree);
void print_forest(struct forest_t forest);
```

5. Да се напише функция за отсичане на стари дървета и засаждане на нови на тяхно място. Тя получава за аргументи гората и възрастта, над която дървото трябва да бъде отсечено. Отсечените дървета се заменят с нови с възраст 1 и височина 0.5. Да извежда на екрана съобщение за отсеченото дърво.

```
struct forest_t cut_old_trees(struct forest_t forest, int
age_treshold);
```

6. Да се напише функция за отсичане на високи дървета и засаждане на нови на тяхно място. Тя получава за аргументи гората и височината, над която дървото трябва да бъде отсечено. Отсечените дървета се заменят с нови с възраст 1 и височина 0.5. Да извежда на екрана съобщение за отсеченото дърво.

```
struct forest_t cut_tall_trees(struct forest_t forest, float
height_treshold);
```

7. Да се обработят следните конзолни аргументи в реда в който ще бъдат подадени на програмата:
    a. Брой цикли, които да се извършат - `int`
    b. Възраст, над която дърветата да се отсичат - `int`
    c. Височина, над която дърветата да се отсичат - `float`

    Пример:

    `./a.out 1000 50 50.456`
8. Да се създаде гора, която да се залеси. Да се направи цикъл до подадения на програмата цикли (в примера - 1000), като във всяка итерация:
    a. Да се изпълни `age_trees` върху гората
    b. Да се изпълни `cut_old_trees` върху гората, като се използва възрастта за отсичане (в примера - 50)
    c. Да се изпълни `cut_tall_trees` върху гората, като се използва височината за отсичане (в примера - 50.456)

    Преди и след цикъла да се изведе информацията за цялата гора.

Задача 3.

***В тази задача ще се имплементира силно опростен вариант на алгоритъма за криптиране Blowfish. Тъй като все още не сме учили указатели, няколко от функциите не могат да върнат масив, или да променят стойностите на L и R. Затова като последен аргумент получават char*, в който могат да пишат стойности, които да се използват от извикващата функция. Както и досега, ще приемем, че `char* var` и `char var[дължина]` са равнозначни.**

1. Да се дефинира глобалната променлива S
   a. `char S[2][16]` - двумерен масив от символи

   Да се дефинира функция F, която получава аргумент символ и връща резултата от `S[0][символът отместен 6 бита надясно] + S[1][символът отместен 4 бита надясно]`.

   ```
   char F (char x);
   ```

2. Да се дефинира функция за криптиране на 2 символа. Аргументите, които получава, са символите L и R и масива out, в който ще запише променените стойности на L и R. Функцията трябва да имплементира следната модификация на алгоритъма:

   На R се присвоява стойността от R XOR F(L). На L се присвоява стойността на L XOR F(R). Това се повтаря 16 пъти. След това стойностите на L и R се разменят. На 0-вия елемент на out се присвоява стойността на L, а на 1-вия елемент се присвоява стойността на R (по този начин на практика тези модифицирани стойности стават резултата, върнат от функцията).

   ```
   void encrypt(char L, char R, char* out);
   ```

3. Да се дефинира функция за декриптиране на 2 символа. Аргументите, които получава, са символите L и R и масива out, в който ще запише променените стойности на L и R. Функцията трябва да имплементира следната модификация на алгоритъма:
   На R се присвоява стойността от R XOR F(L). На L се присвоява стойността на L XOR F(R). Това се повтаря 16 пъти. След това стойностите на L и R се разменят. На 0-вия елемент на out се присвоява стойността на L, а на 1-вия елемент се присвоява стойността на R (по този начин на практика тези модифицирани стойности стават резултата, върнат от функцията).

   ```
   void decrypt(char L, char R, char* out);
   ```

4. Да се напише функция за инициализация на S, която получава като аргумент ключа. Всеки елемент от S да се инициализира чрез елемент или елементи от ключа, като тук ви даваме свобода на фантазията.
   Най-прост вариант - на всеки елемент от S да се дава стойността на пореден елемент от ключа, а ако дължината на ключа е по-малка от 32 при достигане на края му да се започва отново от началото.

   ```
   void init_boxes(char* key);
   ```

   Пример:
   ключ => asdf
   S[i][16] =>
   i=0 => a s d f a s d f a s d f a s d f
   i=1 => a s d f a s d f a s d f a s d f
5. Да се напише функция за криптиране на текст с аргументи текста, ключа, и стринга, в който ще бъде записан криптирания текст. Функцията трябва да извика `init_boxes` с ключа, след което за всеки 2 символа на текста да ги

подаде на `encrypt`. `encrypt` ще запише криптираните символи в `out`, от който трябва да бъдат взети и записани в `encrypted`.

```
void perform_encryption(char* text, char* key, char* encrypted);

text -> L, R -> encrypt -> out -> encrypted
```

6. Да се напише функция за декриптиране на текст с аргументи криптирания текст, и стринга, в който ще бъде записан декриптирания текст. Подобно на `perform_encryption` функцията ще подава по 2 елемента на `decrypt`, което ще записва декриптираните символи в `out`, от който трябва да бъдат взети и записани в `decrypted`.

```
void perform_decryption(char* encrypted, char* decrypted);

encrypted-> L, R -> decrypt -> out -> decrypted
```

7. В main да се вземат текста и ключа, които ще бъдат подадени като конзолни аргументи в този ред:
    a. Текст - `char*`
    b. Ключ - `char*`

Текстът няма да надвишава 64 елемента, така че това е достатъчна максимална дължина за променливите за криптиран и декриптиран текст.

Да се демонстрира успешно криптиране на текста.

Да се демонстрира успешно декриптиране на криптирания текста.