Задължително домашно 2

Решението на всяка задача трябва да има makefile. След извикване на make трябва да се генерира изпълним файл zad<номер на задачата>.out.
Решенията трябва да се компилират без грешки под Линукс с

```
gcc -Wall --pedantic --std=c11
```

Решения, които не се компилират, няма да се оценяват.
Решенията да се качват като един архив, в който всяка задача е в отделна папка. Да се качва единствено сорс код без генерирани или изпълними файлове.

1. Да се имплементира функцията:

   ```
   void heapsort(int *array, unsigned int size);
   ```

   която като аргументи получава масив и броя на елементите в него и изпълнява алгоритъма heap sort.

2. Като използвате следната дефиниция за елемент от двусвързан списък:

   ```
   struct node_t {
     int value;
     struct node_t* prev;
     struct node_t* next;
   }
   ```

   имплементирайте функцията:

   ```
   int is_cyclic(struct node_t *node);
   ```

   която връща 1 ако подаденият елемент е член на цикличен списък и 0 ако не е.

3. Като използвате имплементация на стандартно binary search tree с числа (по-малките се поставят в левия клон, по-големите - в десния, повторения не се допускат),
   имплементирайте функцията:

   ```
   void split(struct node_t *tree, struct node_t **result1, struct
   node_t **result2);
   ```

   която разделя елементите от дървото в 2 нови дървета, които връща чрез указателите result1 и result2. Разделянето на елементите става като, ако бъдат взети в сортиран ред, се редува да се поставят в първия и втория резултат.
   За оценка 6 - обхождането и разделянето да не използват допълнителни помощни структури(масив, списък, опашка и т.н.).

Пример:
От дървото

```
      10
    /    \
  5       15
 / \     /  \
3   6   12  18
     \     \
      8     13
```

което има елементи

```
3 5 6 8 10 12 13 15 18
```

елементите се разделят през един

```
3 _ 6 _ 10 _ 13 _ 18
_ 5 _ 8 _ 12 _ 15 _
```

и се получават дърветата

```
    10
   /  \
  6    18
 /    /
3    13


    12
   /  \
  8    15
 /
5
```