## Task 1005

```
/*****************************************************
 * Task 1005                                         *
 *                                                   *
 * SOLVE FOR TIMUS                                   *
 *                                                   *
 *****************************************************/

#include <stdio.h>
#include <stdlib.h>

int get_min_from(int x, int y) { return y ^ ((x ^ y) & -(x < y)); }

int r_func(int iter, int spot1, int spot2, int values[]) {

  if (iter == 0)
    return abs(spot1 - spot2);
  else {
    int to_first_spot =
        r_func(iter - 1, spot1 + values[iter - 1], spot2, values);

    int to_second_spot =
        r_func(iter - 1, spot1, spot2 + values[iter - 1], values);

    return get_min_from(to_first_spot, to_second_spot);
  }
}

int main(void) {
  int iter;

  if (scanf("%d", &iter) != 1) {
    printf("Invalid input");
  }

  int values[iter];

  for (int i = 0; i < iter; i++) {
    if (scanf("%d", &values[i]) != 1) {
      printf("Invalid input: you provided wrong amount of variables");
    }
  }

  printf("%d", r_func(iter, 0, 0, values));
}
```

Здесь основная идея заключалась в том, чтобы сделать обход дерева рекурсивным алгоритмом. Это прекрасная задача, на мой взгляд. Потому что первые мысли, которые приходили на голову - это построить дерево и найти минимальный или считать все данные потом сделать аналитические вычисления. Тем не менее, оказалось, что, грубо говоря, эти два метода можно объединить в один.

Сама функция очень простая - мы просто высчитываем все возможные варианты (образуя некое дерево). Важно при этом в конце каждой функции возвращать абсолютное значение.

С каждого узла дерева мы берем минимальное значение и так далее до корневого узла, откуда нам остается только вернуть минимальное значение.