TASK 1

Что не так с кодом ниже? Объясните и исправьте.

```
class A
{
public:
      A() {}
      ~A() {}
};

class B: public A
{
public:
      B() : A() {}
      ~B() {}
};

int main(void)
{
      A* a = new B();
      delete a;
}
```

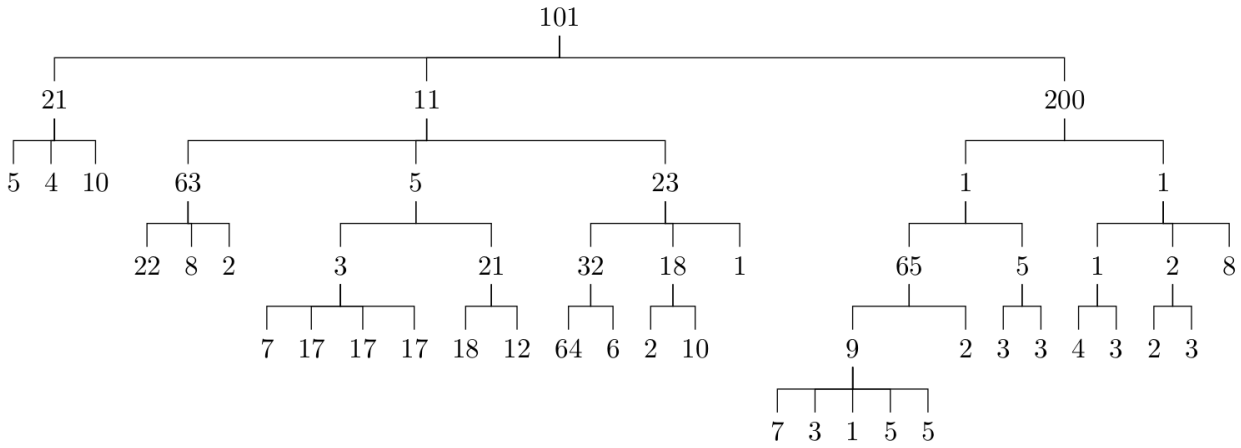
TASK 2

Напишите алгоритм на C/C++ или псевдокоде, который для заданного дерева каталогов вычисляет его общий размер для каждого каталога (рекурсивно). Предлагаемое представление каталога:

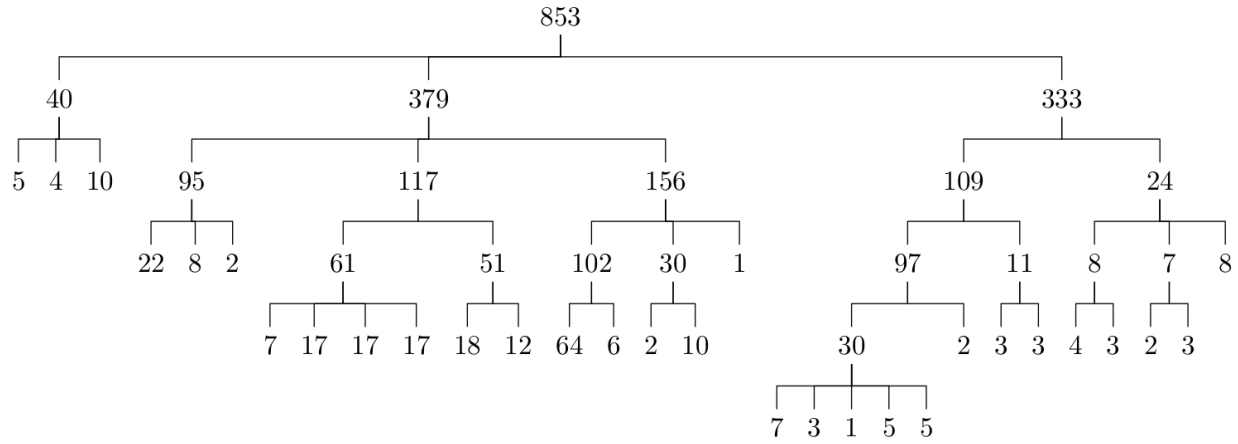
```
struct dir {
      // lista plików znajdujących się w katalogu
      std::vector<file*> files;

      // lista podkatalogów
      std::vector<dir*> dirs;

      // całkowity rozmiar katalogu, który należy obliczyć
      size_t size;
};
```

Предлагаемый прототип функции `compute_dirs_sizes`, для реализации:

```
// Функция, которая берет указатель на корень дерева каталогов, вычисляя
// рекурсивно общий размер каждого каталога


size_t compute_dirs_sizes(struct dir *d) {
      /*
 * Завершите тело функции так, чтобы для каждого каталога d
* d->size — это общий размер каталога d (рекурсивно,
* т. е. включая подкаталоги).
*
* Предположим, что существует метод .size() для `struct file`, который
возвращает
* Размер файла. Например: d->files[0].size() возвращает размер
* первый файл в каталоге d.
*/

}
```



Например, для приведенного выше дерева каталогов, где числа в узлах означают общий размер всех файлов в данном каталоге.(czyli):
результирующее дерево, вычисленное алгоритмом, должно выглядеть так:

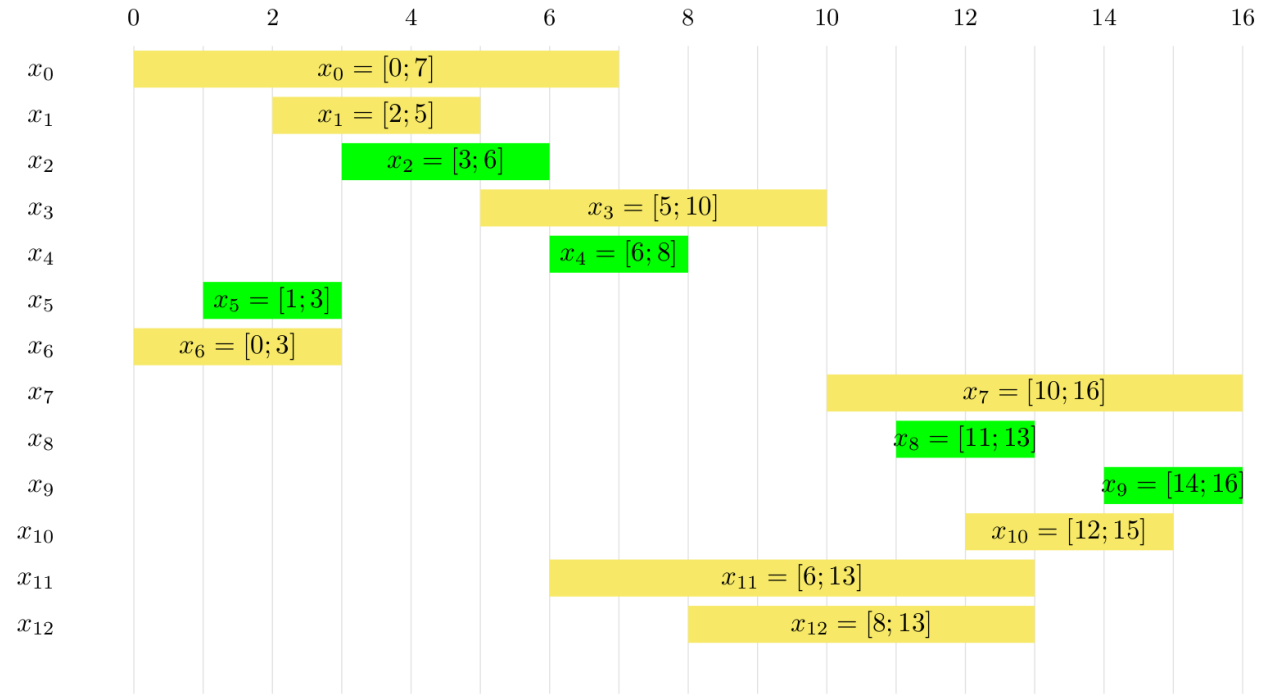


Числа в узлах на рисунке выше представляют общий рекурсивно вычисленный размер каталога со всеми подкаталогами (то есть dir.size). Из приведенного выше дерева видно, что общий размер всех файлов в дереве примеров составляет 853.

TASK 3

В компании есть один конференц-зал и за доступ к нему соревнуются N команд. Каждая команда определяет период времени, в течение которого она хочет забронировать номер. В конференц-зале одновременно может находиться только одна команда. Каждая команда может подать только один запрос, т.е. они могут указать только один период времени, в течение которого они хотят использовать комнату. Напишите алгоритм на C/C++/псевдокоде, который вычисляет, сколько команд могут использовать конференц-зал.

Например, 13 команд и их потребности:
{x_0 = [0;7], x_1 = [2;5], x_2 = [3;6], x_3 = [5;10], x_4 = [6;8], x_5 = [1;3], x_6 = [0;3], x_7 = [10;16], x_8 = [11;13], x_9 = [14;16], x_{10} = [12;15], x_{11} = [6;13], x_{12} = [8;13]}
можно изобразить на следующей схеме:



Для приведенного примера правильный ответ 5. Пример выбора 5 команд, которые получат зал, показан зеленым цветом - это команды: {x2, x4, x5, x8, x9} (это не возможен только выбор из 5 команд).

Укажите вычислительную сложность и сложность памяти предлагаемого алгоритма.

TASK 4

Каково будет значение val после запуска и завершения 5 потоков(), если val используется всеми потоками?

```
size_t val = 0;

void thread()
{
      for (size_t i = 0; i < 10000000; i++)
            val++;
}
```

LIZARDFS

TASK 5

Напишите на C/C++ алгоритм, проверяющий, можно ли отсортировать заданный вектор с помощью одной операции std::swap(). Какова вычислительная сложность и сложность памяти предлагаемого алгоритма?
Np.:

- Для последовательности {1, 5, 3, 4, 2} такая замена возможна — достаточно заменить 5 на 2;
- Для строки {1, 2, 4, 3, 5} такая замена возможна — достаточно заменить 4 на 3.
- Для последовательности {4, 1, 2, 3} такой замены нет.

TASK 6

Имея двухмерную доску и слово, проверьте, можно ли составить слово из букв на доске. Слово можно построить, образуя «змейку» из соседних ячеек, где под соседними мы подразумеваем граничащие друг с другом по вертикали или горизонтали. Вы не можете использовать одну и ту же букву с доски более одного раза (конкретная буква, например, с использованием нескольких разных «A», конечно, совершенно нормально).

Пример:

```
board =
[
   [‘A’, ’B’, ‘C’, ‘E’],
   [‘S’, ‘F’, ‘C’, ‘S’],
   [‘A’, ‘D’, ‘E’, ‘E’]
]
```

Для заданного слова ABCCED вернуть true.
Для заданного слова SEE вернуть true.
Для заданного слова ABCB вернуть false.

TASK 7

Реализуйте на C++ алгоритм, который для заданного вектора std::vector<int> v и int X найдет все пары элементов вектора v, сумма которых равна X. Можно предположить, что все числа в векторе v попарно различны.

Пример:

```
v = [5, 2, 8, -1, 0, 7]
X = 7
```

Результат: `[[5, 2], [8, -1], [0, 7]]`.