Задание 4. Рекурсивная обработка списков

Предисловие

Реализуемые функции должны быть сконструированы при помощи рекурсии и сопоставления с образцом. **Запрещается** использование в этом задании функций, определённых в модуле `Data.List`.

Определения

- **Ассоциативный список (A-список)** — это тип данных, представляющий собой список `[(a, b)]` пар `(a, b)`. Будем называть первый элемент такой пары “ключём”, а второй – “значением”.

Задание 1

Требуется определить функцию `a_zip :: [a] -> [b] -> [(a, b)]`, которая принимает на вход два списка и возвращает список пар, каждая из которых сформирована из двух соответствующих элементов исходных списков. Если поданные на вход имеют различную длину, то итоговый список должен иметь длину наименьшего.

Пример:

```
GHCI> a_zip [1, 2, 3] ["one", "two", "three", "four"] == [(1, "one"),
        (2, "two"), (3, "three")]
True
```

Задание 2

Предлагается определить функцию `a_find :: Eq a => [(a, b)] -> a -> b`, которая в переданном списке пар находит значение, соответствующее заданному ключу. Если в списке имеется несколько пар с таким ключём - вернуть значение из первой такой пары, если в списке нет пары с таким ключем – вычислить `error "no such key"`.

Пример:

```
GHCI> a_find [(1, "one"), (2, "two"), (3, "three"), (1, "ten")] 1 ==
        "one"
True
```

Задание 3

Предлагается определить функцию `a_domain :: Eq a => [(a, b)] -> [a]`, которая принимает на вход ассоциативный список и возвращает список уникальных ключей.

Пример:

```
GHCI> a_domain [("temp", 34), ("height",80), ("weight", 180),
        ("depth", 7), ("height ", 80)] == ["temp", "height",
        "weight", "depth"]
True
```