Shopping

Let's define two structures:

```
struct Item
{
    string name;
    double price;
    Item * pPrev, *pNext; // pointer to previous and next Item in a double-linked list
    // some other data
};

struct Customer
{
    string name, surname;
    Item * pBoughtItems;          // head of list of items
    Customer * pLeft, * pRight; // children of tree node
    // some other data
};
```

The structures are used to build a complicated data structure – binary tree of doubly linked lists – Fig. 2. Customers are gathers in a binary tree ordered by surnames. Items of one customer build a doubly linked list in ascending order of items' prices. A customer may have any length of items, no items as well.

1. Define a function `add` that adds an item (price and name given) to a customer (name and surname given). If a customer is absent, a new customer object is created in the correct localisation. An empty list of items is also possible.

2. Define a function `remove` that removes an item from a customer. Parameters of the function: root of a tree, name and surname of a customer, name and price of an items. The function returns:

   - **true** if item removed successfully,
   - **false** otherwise.

   An empty tree is possible.

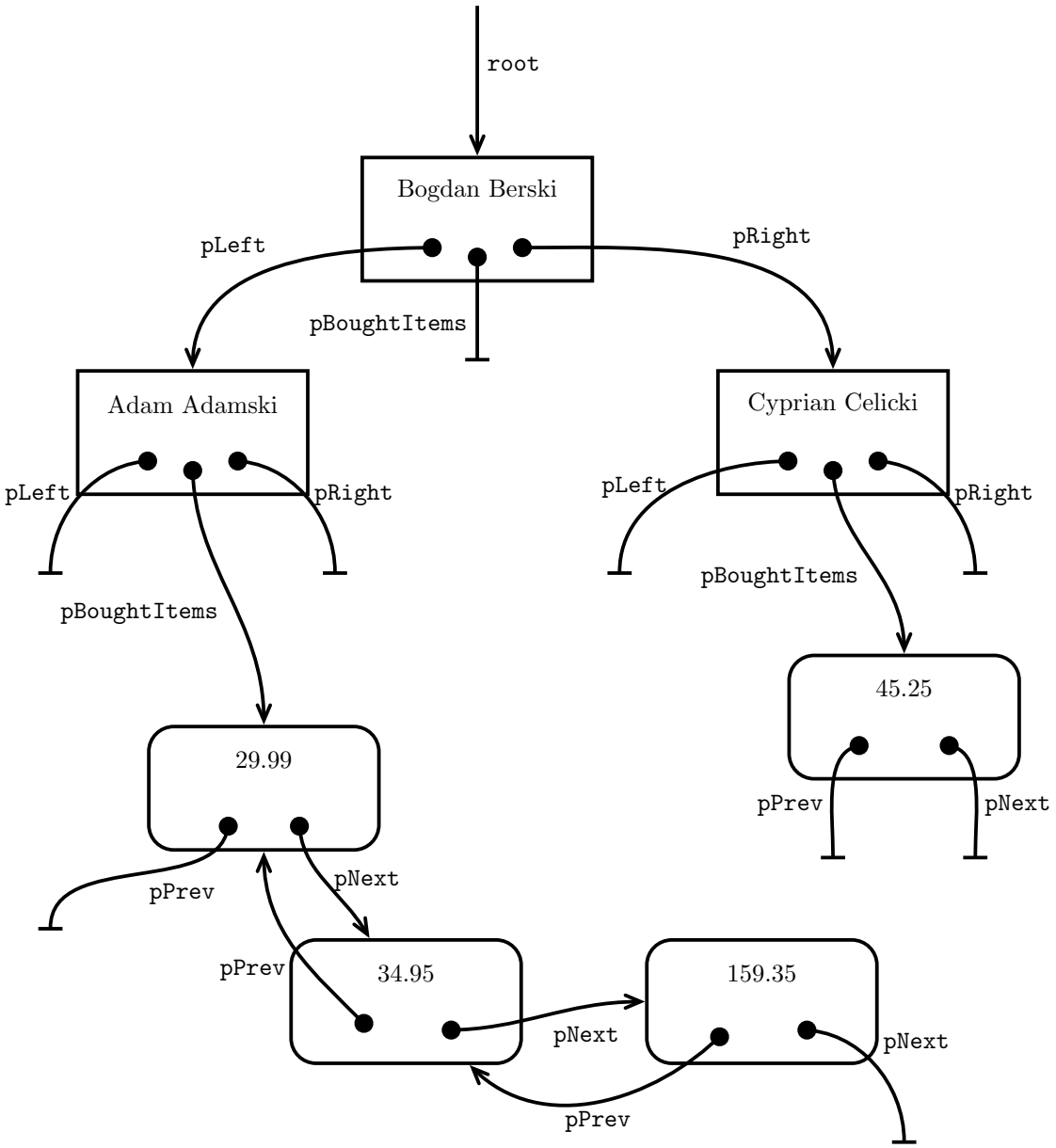

Figure 2: Example of a structure with customers and items.