PJC 05

Zadanie 1

Zdefiniuj strukturę Point z polami x i y (typu double) odpowiadającymi współrzędnym na płaszczyźnie kartezjańskiej. Zdefiniuj też strukturę Rect opisującą prostokąty na płaszczyźnie kartezjańskiej z bokami równoległymi do osi; polami tej struktury są dwa punkty będące lewym-górnym i prawym-dolnym wierzchołkiem prostokąta.

Napisz funkcję

```
std::vector<double> process(const Rect* rects, size_t sz, std::function<double(Rect)>
                                                                    f);
```

która pobiera tablicę prostokątów i jej wymiar oraz funkcję typu Rect→double, a zwraca wektor wyników przekształcenia kolejnych prostokątów z tablicy dostarczoną funkcją.

Napisz program testujący napisana funkcję; jako trzeciego argumentu wywołania użyj zarówno lambd jak i wskaźników do własnych funkcji.

Jako funkcji transformującej możesz, na przykład, użyć funkcji obliczającej pole prostokąta albo długość jego przekątnej. Wtedy dla prostokątów

$$[(0,4),(4,1)], \quad [(-6,3),(6,-2)], \quad [(-7,4),(8,-4)], \quad \text{wynik}$$

powinien być 12, 60, 120 (pola) i 5, 13, 17 (długości przekątnych).

Zadanie 2

Definiujemy jedno wyliczenie i trzy C-struktury

```
enum Banks {PKO, BGZ, BRE, BPH};

struct Account { Banks
    bank; int balance;
};

struct Person { char
    name[20];
    Account account;
};

struct Couple { Person he;
    Person she;
};
```

W funkcji main tworzymy tablicę par (Couple) z danymi, na przykład, takimi

No	He			She		
	Name	Bank	Balance	Name	Bank	Balance
0	Johny	PKO	1200	Mary	BGZ	1400
1	Peter	BGZ	1400	Suzy	BRE	-1500
2	Kevin	PKO	1600	Katy	BPH	1500
3	Kenny	BPH	200	Lucy	BRE	-201

Zdefiniować funkcję o nagłówku

```
const Couple* bestClient(const Couple* cpls, int size,
                         Banks bank);
```

która zwraca wskaźnik do tej pary (Couple) z tablicy przekazanej jako pierwszy argument (o wymiarze size), która ma największą sumę oszczędności jego (he) i jej (she), ale tylko spośród takich par, w których przynajmniej jedno z małżonków ma konto w banku bank. Jeśli żadna z osób nie ma konta w banku bank, to funkcja zwraca nullptr. Nie wolno zakładać, że stan konta jest nieujemny; może być dowolnie duży dodatni i dowolnie duży ujemny. Jeśli dwie lub więcej par spośród tych, które spełniają narzucony warunek ma takie same, największe, oszczędności, to funkcja zwraca wskaźnik do dowolnej z nich. Na przykład program o schemacie

download *BanksPersons.cpp*

```
#include <iostream> enum Banks {PKO,

BGZ, BRE, BPH};

struct Account { Banks
    bank; int balance;
};

struct Person { char
    name[20];
    Account account;
};

struct Couple { Person he;
    Person she;
};

const Couple* bestClient(const Couple* cpls, int size, Banks
    bank) { // ...
}
```

```
int main() { using std::cout; using std::endl;
    Couple cpls[] = { // ...
    };

    const Couple* p = bestClient(cpls, 4, BRE); if (p) cout << p-
    >he.name << " and " << p->she.name
                << ": " << p->he.account.balance + p-
                                >she.account.balance << endl;
    else cout << "No such couple...\n";
} powinien wypisać coś w
```

rodzaju

```
Kenny and Lucy: -1
```

Zadanie 3

Zdefiniowana jest struktura Node:

```
struct Node {
    int  data;
    Node* next;
};
```

Program tworzy listy jednokierunkowe obiektów tego typu. Dla tworzonych list należy zapewnić następującą funkcjonalność:

- dodawanie do listy węzła zawierającego dane typu int w ten sposób, że jeśli na liście jest już węzeł z daną o tej wartości, to nowy węzeł *nie* jest dodawany, a funkcja zwraca false; w przeciwnym przypadku nowy węzeł jest dodawany (na początek listy), a funkcja zwraca true;
- znajdowanie aktualnego rozmiaru (ilości węzłów) listy;
- drukowanie zawartości listy za pomocą funkcji printList;
- usuwanie wszystkich węzłów listy (z drukowaniem informacji o usuwanych elementach) — funkcja clear.

Do funkcji add oraz clean „głowa” listy (a więc wskaźnik na pierwszy węzeł listy) przekazywana jest przez referencję, tak, aby funkcja mogła ją zmienić. Schemat programu:

download *SimpList.cpp*

```
#include <iostream>
struct Node { int data;
    Node* next;
};
bool add(Node*& head, int data); size_t
size(const Node* head); void
```

```
clear(Node*& head); void
printList(Node* head);

int main() { using std::cout; using std::endl; int tab[] = {1,4,1,3,5}; Node*
    head = 0; for (size_t i = 0, e = std::size(tab); i != e; ++i) { bool b =
    add(head,tab[i]);
            cout << tab[i] << (b ? "              " : " NOT ")
              << "added" << endl;
    } cout << "Size of the list: " << size(head) << endl;
    printList(head); clear(head);
}
```

Program powinien wydrukować

```
1       added
4       added
1 NOT added
3       added
5       added
Size of the list: 4
5 3 4 1
DEL: 5 DEL: 3 DEL: 4 DEL: 1
```