Datum: Jméno a Příjmení:.....

1. Uvažujte následující fragment programu. Co bude na standardním výstupu po jeho provedení? [2B]

```
int i, &ri = i;
int *p_i = &i;
i = 5; ri =10;
*p_i = 15;
std::cout << i << " " << ri << std::endl;
```

Řešení:

---

15 15

2. Následující fragment obsahuje několik chyb v indexaci pole. Opravte je. Cílem programu je inicializovat pole ia na hodnoty 1, 2, 3, ... [2B]

```
const size_t array_size = 10;
int ia[array_size];
for (size_t ix = 1; ix <= array_size; ++ix)
  ia[ix] = ix;
```

Řešení:

Indexace by měla probíhat od 0 do array_size-1.

3. Obsahuje následující funkce chybu? Pokud ano, vysvětlete proč. [2B]

```
void square(int i, int *y) {
  int pow;
  pow = i * i;
  y = &pow;
}
```

Řešení:

Vrací referenci na automatickou proměnnou.

4. Popište slovy, co znamenají následující deklarace [2B]:

```
bool (*pf[10])(const string &, int (*)(int,int));

bool &pf(const string &, const string &);
```

Řešení:

- Identifikátor pf je pole o 10-ti prvcích, kterými jsou ukazatelé na funkce s parametry konstantní reference na string a ukazatel na funkci se dvěma parametry int a vracející int a vrací bool.
- Identifikátor pf je funkce, která má dva parametry typu konstantní reference na string a vrací referenci na bool.

5. Mějme následující deklarace:

```
double calc(double);
int count(const string &, char);
int sum(vector<int>::iterator, vector<int>::iterator, int);
vector<int> vec(10);
```

Která z následujících volání se nepřeloží? A proč? [4B]

```
(a) calc(23.4, 55.1);
(b) count("abcda", 'a');
(c) calc(66);
(d) sum(vec.begin(), vec.end(), 3.8);
```

Řešení:

(a) Calc má jen jeden parameter.

(b) OK

(c) OK

(d) OK

6. Mějme tuto funkci: [5B]

```
void nacti_a_secti_cisla(){
    std::cout << "Kolik cisel mam nacist?\n";
    int how_many;
    std::cin >> how_many;
    if (std::cin.bad()) {
        // Nedostali jsme cislo
        std::cout << "To nebylo cislo!\n";
        return;
    }
    int* numbers = new int[how_many];
    for (int n = 0; n < how_many; ++n){
        std::cin >> numbers[n];
    }

    if (std::cin.bad()){
        // chyba nekde v nacitani
        std::cout << "Spatny vstup!\n";
        return;
    }

    // processing ...
    std::cout << "median zadanych cisel: " << median << '\n';
    std::cout << "prumer zadanych cisel: " << average << '\n';
    // ...
    delete[] numbers;
}
```

Jaké chyby v ní vidíte? Jak by jste tuto chybu opravili? Zabrání vaše oprava opakovanému výskytu této chyby?

Řešení:

- Chybí delete před return.
- Doplnit delete.
- Ne, bylo by zapotřebí použít unique_ptr nebo vector.

7. Mějme tuto třídu: [3B]

```
class foo {
    int* p;
    int sz;
public:
    foo(int size):sz(size), p(new int[sz]){}
    foo(const foo& rhs) = default;
    foo& operator=(const foo& rhs) = default;
    foo(foo&& rhs) = delete;
    foo& operator=(const foo&& rhs) = delete;
    ~foo(){delete p;}
    …
    …
};
```

Je takováto deklarace v pořádku? Proč?

Řešení:

- Kopírující konstruktor okopíruje hodnotu pointeru, takže se destruktory různých objektů pokusí uvolnit stejnou paměť.