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 = 15;
*p_i = 10;
std::cout << i << " " << ri << std::endl;
```

Řešení:

```
10 10
```

2. Je dána funkce:

```
void secti(int a, int b, int c) { c = a + b; }
```

Upravte hlavičku funkce tak, aby ji bylo možné volat níže uvedeným způsobem a do proměnné **v** se uložil výsledek výpočtu. [2B]

```
int main()
{
      int c1 = 5, c2 = 4, v;
      secti(c1, c2, v);
      return 0;
}
```

Řešení:

```
void secti(int a, int b, int &c)
```

3. Je dána funkce

```
void tisk(char a, int b) {
      cout << a << b << a << endl;
}
```

Upravte hlavičku funkce **tisk** tak, aby ji bylo možné volat níže uvedenými způsoby a funkce vytiskla to, co je v komentáři. [2B]

```
int main()
{
      tisk('-');        // -5-
      tisk('.',4);// .4.
      return 0;
}
```

```
void tisk(char a, int b = 5)
```

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

```
int (*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 int.
- Identifikátor pf je funkce, která má dva parametry typu konstantní reference na string a vrací bool.

5. Předpokládejte následující realizaci třídy **Predmet**. Vytvořte s její pomocí třídu **RozvrzenyPredmet**, která bude navíc obsahovat informace o místnosti (číselný identifikátor místnosti, předpokládáme číselník místností). Zajistěte, aby položky třídy **RozvrzenyPredmet** bylo možné získat mimo objekt a bylo zachováno zapouzdření objektu.

```
class Predmet {
private:
  int kod;
  string nazev;
  int semestr;
public:
  Predmet(int k, std::string n, int s) :
     kod(k), nazev(s), semestr(s) {};
  string getNazev() { return nazev; };
  int getSemestr() { return semestr; };
  int getKod() { return kod; };
};
```

(a) (1b) Definujte třídu **RozvrzenyPredmet**, která je potomkem třídy **Predmet** a deklarujte potřebné položky této třídy.

```
class RozvrzenyPredmet : public Predmet {

int misto; // předpokládáme číselník místností

};
```

(b) (2b) Definujte konstruktor třídy **RozvrzenyPredmet**, kterým se nastaví všechny položky objektu. Využijte konstruktor rodičovské třídy **Předmět.**

```
RozvrzenyPredmet::RozvrzenyPredmet(int k,string n,int s,int m):

      Predmet(k,n,s), misto(m) {};
```

(c) (1b) Definujte metodu, které vrátí hodnotu položek vlastní třídě **RozvrzenyPredmet**. Protože tato metoda nemění stav objektu, přidejte vhodný modifikátor, který zamezí tomu, aby metoda mohla měnit stav objektu.

```
int getMisto() const { return misto; }
```

6. Na základě výše uvedené třídy **RozvrzenyPredmet**:

   (a) (1b) Deklarujte ukazatel na objekt třídy **RozvrzenyPredmet** a dynamicky jej vytvořte s hodnotami kód: 11, název: "Matematika", semestr: 3, místnost: 201.

```
RozvrzenyPredmet p = new
                    RozvrzenyPredmet(11,"Matematika",3, 201);
```

   (b) (1b) Proveďte nezbytné operace, které je nutné s dynamicky alokovanou pamětí provést po té, kdy se s objektem přestane pracovat.

```
delete p;
```

7. (2b) Definujte nový význam operátoru << tak, aby připojil k výstupnímu proudu **ostr** textovou reprezentaci objektu typu **RozvrzenyPredmet**.

```
ostream &operator<<(ostream &ostr, RozvrzenyPredmet &p)

ostream &operator<<(ostream &ostr, RozvrzenyPredmet &p) {
      return ostr << p.getKod() << " " << .... ;
}
```