1) zapište fragment hlavičkového souboru sdio.h, který zajišťuje, aby nevadilo jeho opakované vkládání.

2) Funkce swap má za cíl prohodit mezi sebou prvky dvou stejně velkých polí. Má funkce nějaké chyby?

```
void swap/int *p, int *q, int pocet){
    int pom;
    for(;--pocet >=0, pom = *p, *p++ = *q, q++ = pom);
    }
```

nejsou tu odpovědi, ale správně bylo a) = funkce chyby nemá

3) Funkce vyhledává maximální hodnotu z pole celých čísel. Ve funkci je chyba.
Kdy funkce nevrátí maximální hodnotu správně?

```
int maxPole(const int *pole, int pocet){
int hodnota = 0;
for(;--pocet>=0; pole++)
if (*pole > hodnota) hodnota = *pole;
return hodnota;
}
```

chybí odpovědi, ale správně bylo a)

4) Funkce for_each zavolá zadanou funkci s jedním číselným parametrem pro všechna čísla zadaného pole. Jak bude vypadat hlavička této funkce za předpokladu, že tělo funkce je:

```
 void for_each(...){
     int i;
     for(i = 0; i < n; i++) fn(*pole++);

     }
```

a) void for_each (const int pole[], int n, void(*fn)(int));
b) void for_each (const int *pole, int n, void fn(int

5) V implementaciítačové hry TETRIS reprezentujte herní plochu (šachtu) polem 10 x 20 buněk typ char, které obsahují informace o dříve spadlých tvarech. Jak použijete standartní funkci memmove, která zkopíruje zadaný počet bytů ze zadané paměťové adresy na jinou adresu (s korektním ošetřením překrývajících se polí), k implementaci funkce zaplnena, kiterá odstraní n-tý řádek (??? pole směrem dolů) při zaplnění řádku?

```
void zaplnena(char tetris[20][10], int radek){
   memmove (...);
   memset(tetris, 0, 10); //vyprázdní horní řádek
    }
```

a) memmove(tetris + 10, tetris,(radek - 1) + sizeof(*tetris));

b) memmove(tetris + 1, tetris,(radek - 1) + sizeof(*tetris));
c) memmove(tetris + radek + 1), tetris + radek, (20 - radek) + sizeof(*tetris));
d) memmove(tetris + radek; tetris, (radek - 1)*sizeof(*tetris));

6) Předpokládejte následující relizaci třídy Prvek, univerzální spojové struktury a vytvořte její pomocí pomocnou třídu IntPrvek, která bude navíc obsahovat celá čísla.

```
class Prve{
    Prvek *dalsi, *predch;
    public:
    Prvek(){dalsi = predch = this;}
    Prvek *vlozZa(Prvek *za);
    Prvek *vlozPred(Prvek *pred);
    Prvek *vyjmiZa(); //vyjmi a vrat dalsi
    Prvek *vyjmiPred(); //vyjmi a vrat predchozi
    }
```

Třída IntPrvek by měla navíc poskytovat konstruktor IntPrvek a metodu vratHodnotu, která vrátí celočíselnou hodnotu uloženou v osloveném prvku.

```
int Prvek(int h);
int vratHodnotu();
```

7) Implementujte zu takto vytvořených "celočíselných prvků" strukturu typu "seznam celých čísel"

```
Class Seznam ... {
    ...
    public:
    Seznam(); //konstruktor
    ~Seznam(); //destruktor
    bool je Prazdny(); //test na prázdnost seznamu
    int delka(); // vrátí počet prvků v seznamu
    };
```

8) Obohaťte seznam "celých čísel" o běžné operace pro pohyb v seznamu

```
public:
    void prvni() //posune ukazovátko na 1. prvek
    void posledni() //  posune ukazovátko na poslední prvek
    void dalsi() // posune ukazovátko na další prvek
    void predchozi() //posune ukazoivátko na předchozí prvek
```

9) Obohaťte seznam o metody pro manipulaci s obsahem.

```
class Seznam ... {
    public:
    void vloz(IntPrvek *p); //vloží prvek dle ukazovátka
    IntPrvek *cti() // vrátí prvek ze seznamu dle ukazovátka
    IntPrvek *vyjmi() // vyjme prvek ze seznamu dle ukazovátka
    };
```

10)Napište funkci (ne metodu) výpis do zadaného proudupřipíše aktuální obsah ze zadaného seznamu. Výsledkem bude celý stav výstupního proudu.

ostream Vypis(s ostream, s Seznam);