Objektum-Orientált Programozás - Zárthelyi dolgozat

Név: _____

Neptun kód: _____

1. **Mi a különbség a stack, heap és adatszegmens között? Az alábbi kódrészletben mely változók melyik területre kerülnek?**

```
double mydouble = 5.1; //globális/névtér szintű változó
int main(){
      int stringLength = myFunction("something");
      static int initToZero4;
      int* q = new int[5];
      delete[] q;
}
```

2. **Mi a különbség az alábbi két (bal és jobb oldali) program között?**

```
int gOne = 1;
void func(int* pInt);
int main(){
      int nvar = 2;
      int* pvar = &nvar;
      func(pvar);
      std::cout << *pvar;
}

void func(int* pInt){
      pInt = &gOne;
}
```

```
int gOne = 1;
void func(int*& rpInt);
int main(){
      int nvar = 2;
      int* pvar = &nvar;
      func(pvar);
      std::cout << *pvar;
      }

void func(int*& rpInt){
      rpInt = &gOne;
}
```

3. **Az alábbi esetek közül mikor vonatkozik a const magára a pointerre, és mikor vonatkozik a pointer által mutatott változó értékére?**

   a) const int* cp1;
   b) int const* cp2;
   c) int* const cp3;

4. **Milyen 3 lépésben történik a C++ forráskód programkóddá alakítása? Ha két külön forrásfájlban is definiáljuk ugyanazt a változót, mi fog történni? Mit jelent az `extern` kulcsszó?**

5. **Mi történik az alábbi 4 kódsorban, ha d és rd valójában nem YellowDog, hanem Dog típusú változókra hivatkoznak (tfh YellowDog szülőosztálya Dog, és a 4 kódsort külön-külön tekintsük, azaz ha az egyik kivételt dob, tegyük fel hogy a program fut tovább)?**

```
int testFunction(Dog* d, Dog& rd){
      YellowDog* yd1 = dynamic_cast<YellowDog*>(d);
      YellowDog& yd2 = dynamic_cast<YellowDog&>(rd);
      YellowDog* yd3 = static_cast<YellowDog*>(d);
      YellowDog yd4 = static_cast<YellowDog>(*d);
}
```

6. **Mit jelent a publikus, privát és protected öröklés? Mikor melyiket érdemes használni? Ami egy szülőben private, azzal mi történik a származtatott osztályban? És egy a szülőben levő public változóval mi történik privát öröklés esetén?**

7. **Mi történik, amikor olyan típusú objektumot teszünk egyenlővé egy ugyanolyan típusú objektummal, mely típushoz nem definiáltunk copy konstruktort?**

8. **Mik a virtuális függvények? Hogyan hozhatunk létre absztrakt osztályt? Mikor jó ez?**

9. **Az alábbiak közül melyik szignatúra felel meg a move assignmentnek?**

   Kutya(const Kutya&);
   Kutya operator=(const Kutya&);
   Kutya& operator=(Kutya&&);
   Kutya(Kutya&&);

10. **Mire használhatunk template-eket? Az alábbi sort() függvény szignatúráján keresztül magyarázza el és adjon példát, hogyan használjuk!**

```
template <class RandomAccessIterator, class Compare>
void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

11. **Mit csinál az alábbi program (rajzoljon magyarázó ábrát is, úgy könnyebben lehet részpontokat szerezni)?**

```
void unknown(int *p, int num){
      int *q = &num;
      *p = *q + 5;
      num = 1;
}

void hardToFollow(int* p, int q, int
*num){
      *p = q + *num;
      *num = q;
      num = p;
      p = &q;
      unknown(num, *p);
}
main(){
      int *q;
      int trouble[3];
      trouble[0] = 11;
      q = &trouble[1];
      *q = 2;
      trouble[2] = 1;
      hardToFollow(q,    trouble[0],
&trouble[2]);
      cout << *q;
}
```

12. **Milyen célt szolgálnak az ún. design patternek? Ismertesse a Builder, Factory és Singleton design patternek lényegét, és közülük az egyikre adjon vázlatos példa-alkalmazást.**