AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2015.03.06. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

Az első két feladatnál a helyes válasz 1 pont, a hibás -1pont! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.

1. Jelölje, hogy mely kijelentés(ek) igaz(ak) a C++ **`inline`** függvényeire! (2p)

   ☐ Az inline kulcsszó nem jelent semmit.
   ☐ Lehet default paramétere.
   ☐ Teljesen azonos a preprocesszor makrójával.
   ☐ A makróhoz hasonlóan törzsük a hívás helyén beépülhet a kódba, paraméterezés tekintetében függvényként viselkednek.
   ☐ Inline függvény nem tartalmazhat elágazást, vagy ciklust.

2. Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre! (1p)

```
{ int *a = new int[12]; a[2] = 2; };
```

   ☐ A kódrészlet teljesen helyes, bár értelmetlen.
   ☐ Hibás, mert memóriaszivárgás lép fel.
   ☐ Hibás, így lenne helyes a foglalás: `int *a = new(12*sizeof(int))`
   ☐ Hibás, így lenne helyes a foglalás: `int *a = new[12] int;`

3. Adott a következő kódrészlet: (3p)

```
int a, b = 5; double x;
f(a) = 12;              // a értéke 12 lesz
f(b)++;                 // b értéke 6 lesz
f(x) = 3.14;            // x értéke 3.14 lesz
f(std::cout, a) << x;  // a és x értéke megjelenik a std. kimeneten
```

   Készítse el C++ nyelven az **`f()`** függvényt (függvényeket) úgy, hogy a fenti kódrészlet a megjegyzéseknek megfelelő hatást váltsa ki! (Makrót nem definiálhat!)

4. Írjon C++ nyelven **teljes programot**, ami annyiszor foglal le egy 1000 elemű egész tömböt a dinamikus memóriaterületen ahányszor ez sikerül! A program a szabványos hibakimenetre írja ki, hogy hányszor sikerült a foglalás! A lefoglalt memóriaterületet kivételesen nem kell felszabadítania! (A C memóriakezelő függvényeit ne használja!) (4p)

3. Feladat megoldása:

```
int& f(int &r) { return r; }
double& f(double &r) { return r; }
std::ostream& f(std::ostream& os, int i) { return os << i; }
```

4. Feladat megoldása:

```
#include <iostream>
int main() {
    int db = 0;
    try {
      while (true) {
        int *p = new int[1000];
        db++;
      }
    } catch (std::bad_alloc) {
        std::cerr << db << " new sikerült!" << std::endl;
    }
    return 0;
}
```

BBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2015.03.06. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

Az első két feladatnál a helyes válasz 1 pont, a hibás -1pont! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.

1. Jelölje, hogy mely kijelentés(ek) igaz(ak) a C++ **`inline`** függvényeire! (1p)

   ☐ Foglalt kulcsszó, de nem jelent semmit.
   ☐ Teljesen azonos a preprocesszor makrójával.
   ☐ Lehet referencia paramétere.
   ☐ A függvény nem tartalmazhat elágazást, vagy ciklust.

2. Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre? (2p)

```
{ double d0, *d = new double[10]; d = &d0; delete[] d; }
```

   ☐ A kódrészlet teljesen helyes, bár értelmetlen.
   ☐ Hibás, mert memóriaszivárgás lép fel.
   ☐ Hibás, **`d`** nem szabadítható fel a **`delete[]`** operátorral, mivel a **`d`** pointer nem a dinamikus területre mutat.
   ☐ Hibás, mert **`*d`** helyett **`**d`**-t kell írni

3. Adott a következő kódrészlet: (3p)

```
long a = 0, b = 1, c = 3, d = 4;
g(c, 1);          // c értéke 4 lesz
g(c, -2);         // c értéke 2 lesz
b = ++g(a);       // a és b értéke 1 lesz
g(a, b, c) << d;  // a, b, c és d értéke megjelenik a std. kimeneten
```

   Készítse el C++ nyelven az **`g()`** függvényt (függvényeket) úgy, hogy a fenti kódrészlet a megjegyzéseknek megfelelő hatást váltsa ki! (Makrót nem definiálhat!)

4. Írjon C++ nyelven **teljes programot**, ami annyiszor foglal le egy 500 elemű valós tömböt a dinamikus memóriaterületen ahányszor ez sikerül! A program a szabványos hibakimenetre írja ki, hogy hányszor sikerült a foglalás! A lefoglalt memóriaterületet kivételesen nem kell felszabadítania! (A C memóriakezelő függvényeit ne használja!) (4p)

3. Feladat megoldása:

```
long& g(long &r, long i = 0) { return r+= i; }
std::ostream& g(long i, long j, long k) { return std::cout << i << j << k; }
```

4. Feladat megoldása:

```
#include <iostream>
int main() {
    int db = 0;
    try {
      while (true) {
        double *p = new double[500];
        db++;
      }
    } catch (std::bad_alloc) {
        std::cerr << db << " new sikerült!" << std::endl;
    }
    return 0;
}
```