4. FELADATSOR

1. Mit ír ki a következő program?

```
int var = 10;
printf("%d, ", 1 + var % 7 * 2);
printf("%d", var);
```

(A) 8, 10
(B) 7, 10
(C) 5, 10
(D) nem fordul

2. Mit írjunk A és B helyére, ha azt szeretnénk, hogy a ciklus visszafelé lépkedjen végig a sample sztring karakterein? (a sample sztringben természetesen lehetnek ismétlődő karakterek)

```
char sample[] = "aaaabbbcccddd";
int i = A;
while (B)
{
    printf("%c, ", sample[i]);
    --i;
}
```

(A) A: strlen(string) - 1; B: sample[i] != '\0';
(B) A: strlen(string) - 1; B: sample[i] != sample[0];
(C) A: strlen(string); B: i >= 0
(D) A: strlen(string) - 1; B: i >= 0

3. Milyen paraméterátadást használ a C nyelv?

(A) érték szerinti
(B) érték és referencia szerinti
(C) érték és cím szerinti
(D) egyik kombináció sem helyes

4. Melyik állítás hamis?

(A) C-ben a tömbök 0-tól kezdődve indexelődnek
(B) C-ben minden tömb utolsó eleme a '\0' speciális érték
(C) C-ben a return utasítással adhatjuk meg a függvény visszatérési értékét
(D) C-ben a void speciális "típus": nem egy adattípus, hanem a típus nélküliségét jelzi

5. Mit állíthatunk az x változóról?

```
int x = 2; // (*)

void foo()
{
    int x = -10; // (**)
}
```

(A) A (**)-al jelölt definíció felüldefiniálja a (*)-al jelölt definíciót, azaz a 2 érték helyére a -10 íródik be
(B) A (**)-al jelölt definíció elfedi a (*)-al jelölt definíciót, azaz a (**) blokkjában létrejön egy másik x nevű változó -10 értékkel
(C) fordítási hibát okoz: x nevű változót már definiáltunk
(D) futáskor az x változó értéke először 2, majd -10 lesz

6. Melyik sor okoz memóriaszivárgást (memory leak)?

```
(1) int* data = malloc(2 * sizeof(int));
(2) *data = -45;
(3) *(data + 1) = 56;
(4) data = malloc(1 * sizeof(int));
(5) *data = 56;
(6) free(data);
```

(A) 3
(B) 4
(C) 5
(D) 6

7. Mit ír ki a következő kód?

```
const char* str1 = "imperative";
printf("%c", *str1);
```

(A) nem fordul: a * operátor operandusa jobbérték
(B) nem fordul: a az "str1" inicializálása nem megfelelő, * helyett [] kellene
(C) "i"
(D) "imperative"

8. Mit ír ki a következő Python program?

```
def foo():
    global var
    var = 1
    print "value of var: ", str(var)
    var = 2

foo()
print "value of var: ", str(var)
```

(A) fordítási hiba: nem létezik Python-ban "global" kulcsszó
(B) fordítási hiba: a var változó csak a foo() függvény törzsében érhető el
(C) "value of var: 1, value of var: 1"
(D) "value of var: 1, value of var: 2"

9. Mi lesz az eredménye a következő Python programnak?

```
t = (1, 2)
t[0] += 3
```

(A) a t értéke (1, 2, 3) lesz
(B) a t értéke (4, 2) lesz
(C) fordítási hiba: a tuple (rendezett n-es) immutable, értéke nem változtatható meg
(D) fordítási hiba: Python-ban nincs += operátor

10. Python-ban melyik állítás igaz a True or False and False logikai kifejezést tekintve?

(A) értéke False mert az or művelet magasabb precedenciájú mint az and, ezért (True or False) and False-ként fog zárójeleződni
(B) értéke True mert az and művelet magasabb precedenciájú mint az or, ezért True or (False and False)-ként fog zárójeleződni
(C) értéke implementációfüggő hisz az operátorok precedenciája nem definiált
(D) fordítási hibát okoz