1. kérdés

0 / 0.5 pont

Ha nem szakítjuk meg egy rekurzív függvény futását valamilyen módon futás közben (pl. valamilyen feltétel teljesülése esetén), mi történik? (A fordítás semmiféle optimalizációt nem tartalmaz.)

Megadott válasz

- (x) Végtelen ciklust kapunk és sose fejeződik be magától a program, csak felhasználói megszakítással fejezhető be a program futása.

Helyes válasz

- ( ) A stack (verem) túlcsordul és vagy segmentation faulttal vagy stack overflow-val terminal a program.
- ( ) Semmi gondot nem okoz.

2. kérdés

0.5 / 0.5 pont

Válaszd ki a helyes állításokat! (Több is lehetséges, rossz válasz mínusz pont!)

Helyes!

- [x] A header-ben a kódot header guard-dal, ifndef vagy pragma makrókkal zárjuk be. Ez megakadályozza, hogy a header többszöri include-jaiból a többszörös deklarációkból és/vagy definíciókból származó névütközések fordítási hibát okozzanak.
- [ ] Egy if elágazásnál mindenképp meg kell írnunk az else ágat is.
- [ ] Mutató változóval csak a program által foglalt és kezelt memóriaterületre tudunk hivatkozni.

Helyes!

- [x] Egy kódból a következő három lépés során válik futtatható állomány: preprocessing, compiling és linking.

3. kérdés

0.5 / 0.5 pont

Mi az a ***tail recursion*** (farok rekurzió)?

- ( ) Olyan önhivatkozó függvény, mely önhivatkozó struktúrák kezelését implentálja.
- ( ) Olyan önhivatkozó struktúra, mely egynél több önhivatkozást is tartalmaznak, mint például a bináris fa vagy a kétirányú lista.

Helyes!

- (x) Olyan önhivatkozó függvény, melynek a return utasításában van az önmagát meghívó kód, így az átalakítható könnyen ciklussá, melyet a fordítók az optimalizáció során meg is tesznek.
- ( ) Olyan önhivatkozó függvény, melynek implementációjában több helyen is szerepel önmagának a meghívása.

4. kérdés

0 / 0.5 pont

Mit ír ki az alábbi ***print()*** függvény, ha meghívjuk? (Include-k természetesen megtalálhatóak a kódban.)

```
struct S
{
    void (*f)(int*);
    struct S* s;
};

void foo(int *a)
{
    printf("%d", *a);
}

void bar(int* a)
{
    ++*a;
}

void print()
{
    int a = 1;
    struct S s1 = {foo, 0};
    struct S s2 = {bar, &s1};
    struct S s = {foo, &s2};
    s.s->s->f(&a);
    printf("%d\n", a);
}
```

Megadott válasz

- (•) 2

- ( ) 1

- ( ) 12

Helyes válasz

- ( ) 11