Prog2 Házi feladat irányelvek

Házi beadása utolsó laborod kezdete előtt 48 órával.

Házi bemutatás elhúzódhat, de a labor napján mindenképp sor kerül rá. Korábbi beadásra van lehetőség, a laborvezetővel egyeztetve.

Házit mindenképp a 14. labor napján kell bemutatni (nem lehet a pótlási héten a "semmit" pótolni, csak orvosi igazolással). A laborvezető engedhet a házin való javítást is a pótalkalomig, amin az eredetileg kapott pontjaid 125%-át tudod max megkapni (tehát olyanra nem lehet számítani, hogy 1% a bemutatón, 100%-ra javítva a pótlásin)! A pótlási hét csak egy extra lehetőség pontosításra (felfelé kerekítésre).

Mivel a házi a jegy hatodát fogja adni, készüljetek komoly kérdésekre a laborvezetőtől (érdemes elméleti tudással felvértezve indulni rajta).

Házi értékelése kapcsán a következőket vedd figyelembe, ezek alapján áll össze az értékelés:

- A specifikáció értékelése és a feladat funkcionális megoldása. A speci értelmesen, a word sablon szerint kifejtve? A feladat azt teszi-e, amit kell? A felhasználó tudja, mit vár el tőle a program? Az elvárt formátumok le vannak írva? Minden követelmény teljesül? Ad értelmes, a használatot bemutató komponenst/main függvényt/...?

- CPP elvek betartása és hatékonyság. Ami kell, az statikus, konstans, stb.? Tagváltozók láthatósága, getterek/setterek, a setterek validálnak? Nincs indoklás nélküli friend? Függvények ha csak lehet, hatékonyan, konstans/referenciát vesznek át? Bevezet sablonokat, ahol értelme van? Konverziókat jól használja? Az állapotot nem változtató tafüggvényeket konstanssá teszi?

- Az osztályok felépítése: öröklés és tartalmazás szempontjából is. Ha sok helyen C-s, procedurális megoldást alkalmazott az OO megközelítés helyett, akkor tipikusan itt csak pár pontot tudsz elérni (ha totál C-s megoldást adsz be, akkor az eleve max pótbeadás, mivel ez nem a C házi). Helyes hierarchiát épített fel? Jól definiált interfészeket? Jól használja a virtuális tagfv-eket? Megfelelő szinteken vezeti-e be a tagváltozókat?

- Hibakezelés megléte: Exceptionök helyesen és jó helyen dobva. Védekezik-e a hibás felhasználói és egyéb inputok (pl. beolvasott fájl formátumhibái stb) ellen, és azokat jól kezeli-e. Ha csak egy nagy catch-el a tesztkódban elkapja a hibát és nem próbál felépülni belőle, az nem túl elegáns. ("Szeretne újrapróbálkozni?/ Megad másik input fájlt? ...) Ne kapja el a kivételt a komponensen belül, ha nem ott kell, lehet ezelni. Ez tipikus hiba szokott lenni. Saját kivétel osztálya az exceptionből származik? Megfelelő exceptiön leszármazottat használ? (Pl. logic_error és nem csak ős-ős exceptiont dob)

- Mennyire szép a kód? Jól vannak-e a komponensek tagolva? Header és CPP fájlok szeparálva? Megfelelő helyen történik az inicializáció? stb.

- Szóbeli bemutatás

- Dokumentáció megléte és minősége. Követi-e a kiadott sablont, szerepel-e minden fontosabb tervezési elem, döntés leírása (pl. miért statikus/friend/template), a sablon esetén milyen előfeltételezésekkel él, stb.

- Szubjektív laborvezetői elem

A hibák arányos pontlevonásaira példát ad a következő táblázat:

7	Nem kezelt exception (komponens esetén a komponens használatát bemutató tesztprogramban)	exceptionönként
7	Memóriakezelési hiba (leak)	
20	Osztályok használatának mellőzése	Illetve a házi visszadobása javításra! OO megoldás kell, nem C HF. Mindent meg lehet írni procedurálisan, vagy akár assemblyben is, de ez nem az a tárgy.
10	Öröklés és tartalmazás keverése	
5	Felesleges/indokolatlan feature használata	pl. csak azért örököltetni, hogy „legyen”
15	HF checkpointkor érdemi haladást nem ért el a hallgató	lefutnia ugyan még nem kell (ha nem szeretne IMSc pontokat), de a program szerkezete és az algoritmusok főbb lépései legyenek meg
5	Publikus tagváltozók használata	teljesítményigényes programok esetén el lehet tekinteni ettől
10	Öröklés használatának mellőzése, ahol kéne	
5	Nem felhasználóbarát hibakezelés	pl. nem értesítjük a felhasználót a keletkezett hibáról, vagy valamilyen futás idejű programozási hibaüzenettel találkozik
10	Megírt osztály/függvény/… használatának mellőzése	felesleges kódot ne tartalmazzon, helyette mutassa be minden implementált funkcionalitás használatát
2	Célt fel nem fedő elnevezések	pl. néhány betűs értelmetlen kifejezések
4	Konstansok használatának mellőzése, ahol kéne	
2	Rekurzív függvény használata	a lassúság és stack overflow veszélye miatt ne alkalmazzuk, mindent meg lehet oldani rekurzió nélkül
2	Globális változók használata	
5	Makro használata	helyesen: #define helyett const vagy inline függvény

Az iMSC pontszám számítása a fentiek alapján az összes nagyHF-re kapott pontszám 80 feletti részének a fele, max 10.

A házi feladat bemutatása olyan, mint egy mini szóbeli vizsga, amelyen 5, a kódodhoz kapcsolódó feladatot kell tudnod megválaszolni. Minden rontás 10%-kal csökkenti a házi feladatra kapott pontokat. Ha legalább 4 kérdésre nem tudod a választ a sajátként beadott kódod kapcsán,

nyilvánvalóvá válik, hogy más által készített, vagy generált kódról van szó. A házi feladatod 0 ponttal elutasításra kerül, és nem tudsz kreditet szerezni a tárgyból az adott félévben.

Ha valaki a pótbemutatón csak javítani akar, nem muszáj élőben megtenni: a kód alapján offline lehet javítani, ha a laborvezető megengedi.

Figyelem! Ha valaki a normál bemutatón egyáltalán semmit nem küld és mutat be, akkor a pótlási héten csak akkor pótolhat, ha orvosi igazolása van az eredeti bemutató napjára.