NHF pontozás

info-C++ 2021 tavasz

Az alábbi táblázat a laborvezetők munkáját segíti. Nem minden feladatra lehet minden értékelési kritériumot alkalmazni, ezért esetenként el kell/lehet térni ettől.

F/H	Leírás	P
Specifikáció (NHF1	A feladat alapfunkciói értelmesen le vannak írva. Tudjuk, hogy mit várhatunk el a programtól és tudjuk, hogy a program mit és milyen formátumban vár el a felhasználótól. Az elvárt formátumok le vannak írva. Az esetleges korlátok is világosak. A korlátokhoz kapcsolódóan nagyvonalakban a hibajelzések megjelennek.	5p
Terv (NHF2)	OO elvek mentén történő tervezés. Osztályok helyes megválasztása, azok kapcsolatának helyes leírása. Láthatóságok, felelősségek szétválasztása. Heterogén kollekció alkalmazása, ha lehet. Ha lehet, ne legyen mindent tudó Isten-objektum. Osztályok attribútumainak helyes megválasztása, jelölése. Működés algoritmikus bemutatása.	5p
Skeleton (NHF3)	Szerepel-e minden fontosabb osztály, sablon. Látszanak-e a kapcsolatok. Tagfüggvények paraméterezése. Tényleg csak skeleton-e. Nem várunk teljes programot, de nem is fogadjuk el skeletonnak. Helyes hierarchiát épített fel? Jól definiálta az interfészeket?	5p
Végső megvalósítás (NHF4)	OO elvek betartása és hatékonyság. Ami kell, az statikus, konstans, referencia, stb. Tagváltozók láthatósága, getterek/setterek, a setterek validálnak. Nincs indoklás nélküli friend. Hatékonyság: ha csak lehet referenciát, pointert vesz át. Nincs hatalmas adat a stack-ben (lokális tömb). Bevezet sablonokat, ahol értelme van. Konverziókat jól használja. Az állapotot nem változtató tagfüggvények konstansok. Nincs osztálytípusra if, switch, ..	5p
	Az osztályok felépítése öröklés és tartalmazás szempontjából is átgondolt és helyes. Sok C-s, procedurális megoldással itt csak pár pontot lehet elérni (ha totál C-s a megoldás, akkor az eleve pótbeadás, bár ennek már a tervezéskor ki kellett buknia). Jól használja a virtuális tagfüggvényeket? Megfelelő szinteken vezeti-e be a tagváltozókat? Szerep szerinti osztálynevek, változónevek, metódusnevek	5p
	Hibakezelés megléte: Kivételek helyes használata. Védekezik-e a hibás felhasználói és egyéb inputok (pl. beolvasott fájl formátumhibái stb.) ellen, és azokat jól kezeli-e. Ne kapja el a kivételt a komponensen belül, ha nem ott kell kezelni. Ez tipikus hiba szokott lenni. Saját kivétel osztálya az exception-ből származik? Megfelelő leszármazottat használ? (Pl. logic_error és nem csak ős-ős exception-t dob)	3p
	Mennyire szép a kód? Jól vannak-e a komponensek tagolva? Header és CPP fájlok szeparálva? Megfelelő helyen történik az inicializáció? Vannak-e kommentek, melyek leírják a függvények bementét/kimenetét lehetőleg doxygen formátumban.	3p
	Teszt: A tesztek megmozgatják-e az összes komponenst (ebben segít a Jporta covarage tesztje). Kihasználta-e a Jporta adottságait, környezetét (feltölthető teszt fájlok, gtest_lite, stb)?	5p
	Dokumentáció minősége. Igényes-e, teljes-e, inkrementális-e? Tartalmazza-e a visszalépéseket, önrevíziókat, ha volt ilyen? Itt vehető figyelembe a feladat komplexitása is.	4p

Inkrementális dokumentációt várunk, azaz minden beadáskor teljes, a korábbi dokumentációváltozatokat is tartalmazó dokumentációt kell beadni.
Mivel a tervezést/specifikációt csak most tanulják, így tervezési lépések összemosódnak az implementációval. Ezért lehetőség van az első két részfeladat felülvizsgálatára a végső beadásig, de maximum 50% pont adható ezekre, ha csak végső beadáskor jelenik meg ez a két részfeladat a dokumentációban.

A tervezési lépések súlyát jelzik a pirossal kiemelt pontszámok. Ebből az látható, hogy egy feladat specifikációja és tervezése nagyobb súlyú, mint az implementáció. A valóságban ez az arány még nagyobb és a tesztelés súlya is jelentősebb.

Tipikus hibák pontlevonásai

Hiba	**Pont**	**Megjegyzés**
Nem kezelt kivétel	**-4**	
Memóriakezelési hiba (leak)	**-20**	Nem fogadható el a HF
Osztályok használatának mellőzése	**-20**	Nem fogadható el a HF
Öröklés és tartalmazás keverése	**-10**	
Felesleges/indokolatlan feature használata	**-4**	pl. csak azért örököltet, hogy „legyen”
Publikus tagváltozók használata	**-4**	
Öröklés használatának mellőzése, ahol kellene	**-10**	
Nem felhasználóbarát hibakezelés	**-3**	pl. nem értesítjük a felhasználót a keletkezett hibáról, vagy valamilyen futás idejű programozási hibaüzenettel találkozik
Megírt osztály/függvény/… használatának mellőzése	**-4**	felesleges kódot ne tartalmazzon
Célt fel nem fedő elnevezések	**-2**	pl. néhány betűs értelmetlen kifejezések
Konstansok használatának mellőzése	**-4**	
Globális változók használata	**-4**	Kivételes esetben pl. grafikus rendszerhez való kapcsolat, de akkor is inkább dugja el egy osztályba static-ként.
Makró használata const, vagy inline fv. helyett	**-4**	

A pótlási héten a normál bemutatón **szerzett pontoknál max. 25%-al magasabb** eredmény érhető el. Pl. bemutatón valaki 50%-ot ért el, akkor nem kaphat 62,5%-nál nagyobbat a pótlásin bemutatott javításokkal.

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.