HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Gagnaskipan

T- 201-GSKI

Lokapróf

Teacher: Kári Halldórsson

Dagsetning: 11. Júlí, 2016
Tími: 9:00 - 13:00

Hjálpargögn: **Rafræna bók, C++ reference og glærur annarinnar má finna í möppunni data á skjáborði prófstölvunnar**

Nafn: _____

Kt.: _____

Answers can be given in English and/or Icelandic.

1. Dynamic array (15%)

Opnið verkefnið *DynamicArray*.
Það er útfærsla á klasanum *IntVector* sem útfærir kviklegt fylki af heiltölum.
Klárið að útfæra **2** föll: *push_back* (**5%**) og *insert* (**10%**).
push_back bætir nýju staki aftast í listann.
insert bætir nýju staki inn í listann, á staðsetningu *index*, án þess að yfirskrifa það sem var á þeirri staðsetningu fyrir. Ef *index* er utan við mögulegar staðsetningar í listanum, m.v. stökin sem komin eru í hann, er kastað ***IndexOutOfBoundException***.
Ef ekki er pláss í fylkinu fyrir nýja stakið, þarf að stækka fylkið.

2. Singly-linked list (15%)

Opnið verkefnið *SinglyLinkedList*
Þar er útfærsla á klasanum *LinkedList*, sem útfærir eintengdan lista sem notar *template*, tekur inn hvaða klasa sem er.
Klárið að útfæra **3** föll: *headInsert*, *tailInsert* og *headRemove* (**5%** hvert).
headInsert bætir staki við fremst í listann (einnig kallað push_front).
tailInsert bætir staki við aftast í listann (einnig kallað push_back).
headRemove fjarlægir stak fremst af listanum og skilar gildi þess (einnig kallað pop_front).
Ef ekkert stak er í listanum á *headRemove* að kasta villunni ***EmptyException***.
Áætlað úttak má sjá í skránni *ExpectedOutput.txt* í verkefnismöppunni.

3. Doubly-linked list (10%)

Opnið verkefnið *DoublyLinkedList*
Þar er útfærsla á klasanum *StringList* sem útfærir tvítengdan lista af strengjum.
Klárið að útfæra **2** föll: *move_to_pos* og *remove* (**5%** hvort).
move_to_pos færir núverandi staðsetningu á *index*-ta stakið í listanum.
Ef *index* er utan við mögulegar staðsetningar í listnum gerir *move_to_pos* ekki neitt.
remove fjarlægir stakið á núverandi staðsetningu úr listanum, eyðir hnútnum sem var fjarlægður og skilar gildinu sem hann innihélt.
Ef núverandi staðsetning er aftan við listann á *remove* að kasta **InvalidPositionException**.

4. Binary search tree(10%)

Opnið verkefnið *BinarySearchTree*
Þar er útfærsla á klasanum *HighScoreBST* sem útfærir tvíundarleitartré af *HighScore*.
HighScore er gagnaklasi sem hægt er að raða, þar sem samanburðarvirkjarnir <, > og == eru yfirskrifaðir á honum.
Klárið að útfæra **2** föll: *addScore* og *find* (**5%** hvort).
addScore er insert fall sem bætir staki á réttan stað í tréð, m.v. röðun byggða á samanburðarvirkjunum á *HighScore* klasanum.
find ber stigagildið sem leitað er eftir við HighScore::score gildi stakanna í listanum þar til það finnur stak sem jafngildir því sem leitað er að. **Þá er því staki skilað**. Athugið að hér þarf að bera integerinn *scoreValue* við integer gildið fyrir stigin í *HighScore* klasanum. Það er **ekki** verið að bera saman *HighScore* klasa og *HighScore* klasa.

Ef stakið finnst ekki í listanum á *find* að kasta ***NotFoundException***.
Áætlað úttak má sjá í skránni *ExpectedOutput.txt* í verkefnismöppunni.

5. Hash table (10%)

Opnið verkefnið *HashMap*
Þar er útfærsla á klasanum *HashMap* sem útfærir tætitöflu með *template*, tekur við hvaða klasa sem er. Gera má ráð fyrir að á klasataginu *K* (*key* er af því tagi) sé útfært fallið *hash()* og yfirskrifaður *samanburðarvirkinn* ==.
Það má semsagt kalla á *key.hash()*, sem skilar integer og það má gera *if(key == it->key)*, ef *it* er *keyPair**, eða *iterator* yfir lista af keyPair, eins og sjá má gert í << virkjanum.
Klárið að útfæra **2** föll: *insert* og *find* (**5%** hvort).
insert bætir í listann *keyPair*, sem inniheldur *K key* og *T value*, og er lyklað í tætitöflunni á *key*.
find finnur stakið sem hefur lykilinn *K key* í töflunni og skilar gildinu *T value* úr *keyPair* tilvikinu sem finnst. Það skilar semsagt bara *value*, en **ekki** öllu *keyPair*.
Ef ekkert stak finnst sem hefur lykilinn *key* á *find* að kasta **NotFoundException**.

6. Erfðir og STL (15%)

Opnið verkefnið *ParkingLot*
Útfærið klasann *Car* sem yfirskrifar << *virkjann*. Það notar *virkjann* til að kalla í föll sem skrifa út á straum nafn bílsins (klasans). Útfærið einnig klasana *Toyota* og *Subaru* sem erfa *Car*. Útfærið þvínæst *Corolla* og *Yaris* sem erfa *Toyota* og *Forester* sem erfir *Subaru*. Hver þessara klasa á að skrifa út nafn foreldris síns, ásamt eigin nafni til viðbótar. Þannig skrifar *Corolla* út "**Car: Toyota Corolla**" á meðan *Toyota* skrifar út "**Car: Toyota**" en *Car* skrifar bara út "**Car**".
Hver klasi á þó bara að skrifa út sinn hluta nafnsins en láta foreldrið um að skrifa út byrjunina.
Ef illa gengur að forrita þessa virkni gangið þá bara úr skugga um að hver tegund skrifi eitthvað ólíkt þegar kallað er á << *virkjann*.
Útfærið núna klasann *ParkingLot*. Hægt er að bæta tilvikum af taginu *Car** inn í *ParkingLot* gegnum fallið *add*. Athugið að **bendar** eru notaðir þannig að STL gagnagrind sem notuð er til að halda utan um bílana þarf að vera safn af *Car** (**<Car*>**) en ekki bara *Car*. Yfirskrifið << *virkjann* í *ParkingLot* og látið hann fara gegnum alla bílana sem bætt hefur verið á bílastæðið og senda þá á strauminn með <<. Ef þeir eru einfaldlega sendir sem *Car** ættu þeir sjálfkrafa að skrifa út, hver sína útgáfu af framleiðanda og tegund, ef erfðirnar hafa verið rétt útfærðar.
Áætlað úttak má sjá í skránni *ExpectedOutput.txt* í verkefnismöppunni.

7. Spurningar (25%)

Spurningarnar eru á linknum Exam á skjáborði próftölvunnar.