III. (25%) Set

Code::Blocks project og öll gögn sem tengjast þessu verkefni má finna í möppunni 3_set. / *Code::Blocks project and everything else related to this project can be found in the directory 3_set.*

Í þessu verkefni eigið þið að útfæra klasann Set sem geymir mengi af heiltölum í tvíleitartré. Hnútar tvíleitartrésins eru tilvik af klasanum TreeNode. / *In this problem you will implement the class Set which stores a set of integers in a binary search tree. The nodes of the binary search tree are instances of the class TreeNode.*

Eftirfarandi skrár eru gefnar. / *You are provided with the following files.*

➜ TreeNode.h Inniheldur yfirlýsingu og útfærslu á TreeNode. / Contains the declaration and implementation of TreeNode.

➜ Set.h Inniheldur yfirlýsingu á klasanum Set. / *Contains the declaration of the class Set.*

➜ main.cpp Inniheldur dæmi um aðalforrit sem notar Set klasann. / *Contains an example of a main program that uses the Set.*

➜ output.txt Inniheldur rétt úttak úr aðalforritinu. / *Contains the correct output of the main program.*

Útfærið klasann Set í skránni Set.cpp. Þið megið bæta við hjálparföllum og breyta yfirlýsingunni á klasanum Set og TreeNode. Þið megið þó ekki breyta yfirlýsingu fallanna sem þið eigið að útfæra. / *Implement the Set class in the file Set.cpp. You are allowed to use helper functions and also allowed to change the declaration of Set and TreeNode. You are, however, not allowed to change the declaration of the functions you are supposed to implement.*

Föllin sem útfæra á í Set, og vægi þeirra, eru eftirfarandi. Munið að setja tímaflækju fallanna inn í Set.h ef við á. / *The functions you should implement in Set, and their weight, is as follows. Remember to add the time complexity of the functions to Set.h when specified.*

1. (2%) Set() (smiður / *constructor*)
Smíðar tómt mengi. / *Constructs an empty set.*

2. (4%) ~Set() (eyðir / *destructor*)

3. (4%) void insert(int n)
Bætir n í mengið. Ef n er til staðar í menginu fyrir, þá hefur kall á fallið engin áhrif. / *Adds n to the set. If n is already present in the set, the function call has no effect.*

i. (3%) Útfærsla. / *Implementation.*
ii. (1%) Hver er versta tímaflækja fallsins, ef við gerum ráð fyrir að tréð sé „balanced?“ / *What is the worst-case time complexity of this function, assuming the tree is “balanced?”*

4. (4%) void print()
Prentar stök mengisins út á staðalúttak í vaxandi röð. Stökin eru aðskilin með einu bili. / *Prints the elements of the set to the standard output in increasing order. Elements are separated by a single space.*

i. (3%) Útfærsla. / *Implementation.*

ii. (1%) Hver er versta tímaflækja fallsins, ef við gerum ráð fyrir að tréð sé „balanced?“ / *What is the worst-case time complexity of this function, assuming the tree is “balanced?”*

5. (5%) int range()

Skilar mismuninum á stærsta og minnsta staki mengisins. Þ.e. skilar Max - Min, þar sem Max er stærsta stak mengisins og Min er minnsta stak mengisins. Ef engin slík stök eru til er TreeException kastað. / *Returns the difference between the largest element of the set and the smallest. I.e., returns Max - Min, where Max is the largest element in the set and Min is the smallest element in the set. If no such elements exist, TreeException is thrown.*

i. (4%) Útfærsla. / *Implementation.*

ii. (1%) Hver er versta tímaflækja fallsins, ef við gerum ráð fyrir að tréð sé „balanced?“ / *What is the worst-case time complexity of this function, assuming the tree is “balanced?”*

iii. (2% bonus) Útfærið fallið með bestu mögulegu tímaflækju. / *Implement the function with the most optimal time complexity possible.*

6. (6%) void remove_min(unsigned int k)

Fjarlægir k minnstu stökin úr menginu. Ef k er stærra en fjöldi staka í menginu, þá eru öll stök mengisins fjarlægð. / *Removes the k smallest elements from the set. If k is larger than the size of the set, all elements of the set are removed.*

i. (5%) Útfærsla. / *Implementation.*

ii. (1%) Hver er versta tímaflækja fallsins, ef við gerum ráð fyrir að tréð sé „balanced?“ / *What is the worst-case time complexity of this function, assuming the tree is “balanced?”*

V. (10%) Tic-tac-toe

Code::Blocks project og öll gögn sem tengjast þessu verkefni má finna í möppunni 5_tictactoe. / *Code::Blocks project and everything else related to this project can be found in the directory 5_tictactoe.*

Í þessum hluta fáið þið gefna útfærslu á leik sem leyfir tveimur leikmönnum að spila Myllu. Leikmennirnir eru ýmist mennskir eða tölvustýrðir. / *In this part you are given an implementation of a game that allows two players to play Tic-tac-toe. The players can either be human or computer.*

Leikurinn er útfærður að fullu. Þið þurfið ekki að bæta neinni virkni við leikinn. Ykkar markmið er að bæta hönnun leiksins. Klasinn Player er notaður til að tákna leikmann. Hann hefur að geyma meðlimabreytuna human, sem segir til um hvort að um sé að ræða mennskan leikmann eða tölvuleikmann. / *The game is fully implemented. You do not need to add any functionality to the game. Your goal is to improve the game's design. T he class Player is used to represent a player. It has a member variable human that tells whether this player is a human player or a computer player.*

Þegar þessar upplýsingar eru geymdar inni í Player klasanum og kallað er á make_move, þá er hegðun leikmannsins ákveðin þar. Þessu viljum við breyta. Við viljum láta fjölvirkni sjá um að ákveða hegðun make_move fyrir okkur. / *When this information is stored inside the Player class and the function make_move is called, the behaviour of the player is decided inside the function. This is what we want to change. We want to use polymorphism to decide the behaviour of make_move.*

Bætið við tveimur klösum, HumanPlayer og ComputerPlayer sem annars vegar útfærir virkni mennskra leikmanna og hins vegar tölvuleikmanna. Notið þessa klasa svo til að breyta hönnun forritsins þannig að fjölvirkni ákveði hegðun make_move. / *Add two classes to the project, HumanPlayer and ComputerPlayer, which implement the behaviour of a human player and a computer player, respectively. Use these two classes to change the design of the program such that polymorphism decides the behaviour of make_move.*

Eftir að þið hafið breytt forritinu, þá á Player ekki að hafa meðlimabreytuna human. / *After you have changed the program Player should not contain the member variable human.*

Athugið að þið þurfið ekki að bæta við nýjum h- og cpp-skrám. Þið megið bæta nýju klösunum við í Player.h og Player.cpp. Athugið einnig að þið megið aðeins breyta main-fallinu í main.cpp (og hugsanlega bæta við include-skipunum). / *Note that you do not need to add any h- or cpp-files. You can declare and implement the new classes in Player.h and Player.cpp. Furthermore, note that you are only allowed to change the main-function in main.cpp (and possibly add include directives).*

Eftirfarandi skrár eru gefnar. / *The following files are given.*

➔ TicTacToe.h/TicTacToe.cpp Inniheldur yfirlýsingu og útfærslu klasans TicTacToe. Þið þurfið ekki að breyta neinu í þessum skrám. / *Contain the declaration and implementation of the class TicTacToe. You do not need to modify anything in these files.*

➔ Player.h/Player.cpp Innihalda yfirlýsingu og útfærslu klasans Player. / *Contains the declaration and implementation of the class Player.*

➔ main.cpp Inniheldur aðalforritið. / *Contains the main program.*