Introduction

For this assignment, you will work towards implementing a simple board game. This is not an interactive game you can play, but rather a simulation of two (or maybe more) players taking turns over a set number of rounds. Play is to be automated - there is no artificial intelligence, and no user input. All output will be text based and directed towards the console – there are no graphical elements to this assignment. In the game, motivated students grind their way inexorably through a computing degree at Scumbag College\*.Along the way they tackle assessments and have various other misadventures. Success means progression to the next year of the course. Lack of success risks dropping out or even worse - being stuck in a cycle of Mitigating Circumstances, endlessly repeating the year. Will any student graduate? Find out in “Degrees of Hell”. I expect you to complete this project in your own time outside of scheduled labs.• This is an individual project and no group work is permitted.• Do not diverge from the assignment specification. If you do not conform to the assignment specification then you will lose marks.

Overview Although there is no graphical element to the program, it may be helpful for you to think of the game as being played on a board comprising 36 spaces arranged in a circle. Each of the spaces has a name (and some other information) which is contained within the data file degrees.txt. In your simulation, the game should (initially) have two players. Player 1 is called Vyvyan, and player 2 is called Rick. At the start of the game, the pieces are placed on the “Welcome Week” space. Players also have a year of study, a level of motivation, and a measure of success. Each player starts the game with their year = 1, motivation = 1,000 and success = 0. During the game, players are allowed to have negative motivation, but not negative success .A game is played as a series of rounds. During a round, each player takes one turn to spin the spinner. You will represent spinning the spinner by generating a random number between 1 and 10 (using the code I have provided – see appendix). The player then moves their piece clockwise round the board by the number of spaces indicated by the spinner, Different things will happen according to the space landed on, which may result in the player gaining or losing motivation and/or success. Exactly what happens on each space depends on how many marks you want.

**Program Specification**

You should implement the features described below in order. To be eligible for a mark within any classification, you must have at least attempted all the features for all the previous classifications. Your code must follow the style guide precisely, and also be well commented throughout.

• Create a class called CSpace to represent the spaces on the board.• Read in the data file degrees.txt (see appendix on the last page) and use it to set up an array or (better) a vector for the 36 CSpace objects.

o You may store the CSpace objects directly (i.e. statically) in the array or vector, but if you do this, your mark will be capped at 40%.

▪ For a mark above 40, the array or vector should store pointers to spaces.

o Each line in the file represents one space.

▪ The first number in each line represents the space’s type, but for a bare pass, you can treat all spaces the same.

▪ Following the type number is the space’s name. You will need to store the name as data in the CSpace objects as you create them.

• Create a class called CPlayer to represent the players. For each player you need to store their name, motivation , success and position on the game board. Start by simulating playing 20 rounds of the game in your main program, with the players simply moving round the board as follows:

• When the game is started, a welcome message is displayed. The format of the message is:'Welcome to Scumbag College'• At the start of each round, the round number should be output:'round <round number>'

• For each turn, the player’s name and the number they spin is displayed. Output the message:'<Player> spins <number>'

• On the next line the name of the player and the name of the space that the player landed on are displayed.Output the message:'<Player> lands on <space name>'

• When a player passes over or lands on the Welcome Week space, the player’s motivation is increased by250. The players year should also be incremented. You must output the message:‘<Player> attends Welcome Week and starts year <year> more motivated! 'When the code runs, there should be no user interaction. Specifically, there should be no need to press a key in-between rounds. You may wish to introduce such a feature to help with development/debugging, but it should be disabled in the code/demo you submit.

**Pass Mark = third classification (40% + )**

• Declare a class to implement the type 1 spaces on the board. These spaces represent assessments.

o An assessment is A Kind Of space.

▪ The assessment class should be derived from the space class.

o In the degrees.txt file, rows beginning with ‘1’ denote assessment spaces. Set up the correct objecttypes (assessments or general spaces) as you read in the file.

o As well as their type and their name, assessments also have a motivational cost, and anachievement.

▪ They also have a year, but you can ignore the year at this stage.

o The motivational cost and success score (as found in the degrees.txt file) are given in the table below:

A table of a test

Description automatically generated with medium confidence

• When a player lands on an assessment space the following should happen:

o If the assessment has not been completed by anyone, the player who landed on the assessment will complete it if and only if they have enough motivation to do so.

▪ If completed, the motivational cost of the assessment is deducted from the player and the achievement score is added to the player’s success. The player also records that assessment is complete.

o If the assessment is already completed by the player who landed on it, nothing happens.

o If the assessment has already been completed by another player, the player who landed will ask that other player for help with the assessment.

▪ The motivational cost to the player who landed is halved, and the achievement score is divided, with half going to each player.

o If the assessment is completed by the player on this turn, then output the message:

'<Player> completes <assessment> for <cost> and achieves <score>’

o If the another player helps with the assessment: '<Player> helps and achieves <score>’

• For all the other spaces (except the Welcome Week space – whose row in the file begins with ‘2’ – and where the players collect motivation) nothing should happen when a player lands on them except outputting their name.

• At the end of each player’s turn you must output the message: '<Player>’s motivation is <motivation> and success is <success>’

• At the end of the game you should output the final success for each player, and also which player has won. The winner is the player with the greatest success. Output the following messages: 'Game Over''<Player1> has achieved <success>'<Player2> has achieved <success>‘<Player> wins.’

**Lower second classification (50% + )**

• For this grade you need to attempt a polymorphic solution.

o You will need a hierarchy of classes for the different space types.

o When it comes to implementing the different behaviour of the spaces, the functionality is devolved to the lower levels of the hierarchy:

▪ You should have a collection of pointers of type CSpace\*, but the bespoke implementation code should be written in the derived classes.

▪ You should not call methods on the derived classes directly.

• You must use an STL container to store/access the CSpace pointers (arrays no longer allowed!).

• Implement additional classes derived from CSpace for the Plagiarism Hearing space, Accused of Plagiarism space and Skip Classes space (types 6, 7 and 8 in the text file).

o If a player lands on the 'Plagiarism Hearing' space after spinning, the player is considered to be 'moral support' for their friend, and nothing special happens. Output the message:'<Player> lands on Plagiarism Hearing and supports their friend'

o If a player lands on “Accused of Plagiarism” then their piece immediately moves to the “Plagiarism Hearing” space and their motivation is reduced by 50. Output the message:'<Player> lands on Accused of Plagiarism and goes to the hearing''<Player> loses motivation

o The turn after a player has been moved to Plagiarism Hearing is treated as a regular turn.

• If a player lands on "Skip Classes" then nothing happens. Output the message:'<Player> lands on Skip Classes and hangs out with their dodgy mates'

• Implement a further derived class for Extra-Curricular spaces (type 3).

o The rules for undertaking extra-curricular activities are the same as for completing assessments.

▪ The motivational cost of undertaking any extra curricular activity is 100▪ The amount of success gained from undertaking any extra curricular activity is 20

o If the extra-curricular activity is undertaken by the player on this turn, then output the message:'<Player> undertakes <activity> for <cost> and achieves <score>’o When landing on an activity which has been undertaken by another player, the motivational cost is halved (as with assessments) but other player should receive the same (halved) amount of motivation in addition to sharing the success. The following message should be output:'<Player> motivates <receiving player> by joining their activity'

**Upper second classification (60% + )**

• For this grade, the polymorphic aspect of your solution must be fully correct, and the polymorphism must be serving a genuine purpose.

o You will not be awarded marks if you just plonk the keyword "virtual" into your code, whilst the work within your code is still done in a procedural fashion.

• Derive additional classes from CSpace to implement the Bonus and Bogus spaces.

• If a player lands on either the Bonus or Bogus spaces, the spinner must be spun again, and action taken according to the following table: Spin Bonus (Gain Motivation) Bogus (Lose Motivation)1 Receive some useful feedback. 20 Fall asleep in a boring lecture. 202 Win a hackathon. 50 Meet with coach about poor attendance. 503 Get a free coffee with full loyalty card. 80 Feel really tired after a night out. 804 Impress your lecturer in class. 100 Get kicked out of lab for messing about. 1005 Motivational talk from course leader 5 Get dumped before a big night out. 1506 Secure an industry placement 150 Fail an assignment. 2007 Attend an inspiring C++ lecture. 200 Get caught skipping class. 508 Get your best ever assignment mark 300 Your friend drops out of uni. 2009 Make a new romantic friend 150 Nobody turns up to teach your class. 30010 Get elected School President 10 Lecturer changes assignment schedule. 20

• Output the following including the bonus / bogus message as given above, e.g.'<Player> lands on Bonus’' Win a hackathon. Gain motivation of 50''<Player> has <current motivation>'• At this level there should be no global variables in your code. (Global constants are allowed.)

• There must be no memory leaks when you run your code.

o Proof of this should be included in the document/video demonstration.

**First classification (70% + )**

• Assessments belong to years (as read in from the file)

o When a student reaches Welcome Week, they should only be allowed to progress to the next year of study if they have completed 3 assessments for the year they are just finishing.

▪ They may still complete assessments for other years and gain success.

o If a student doesn’t progress to the next year, they simply go round again and repeat the current year until they meet the criteria to progress.

o If a student successfully completes the 3rd year, they have won the game (they Graduate) , and the game should end immediately. Output a message congratulating them on their graduation.

• The implementation should make use of object-oriented methods throughout (not just the polymorphic parts). This will mean the implementation of several classes.

o One of the classes must be the “Degrees” game itself, which will act as a manager class.

o All your classes should have their own header file and source file.

• The STL syntax for vectors is ugly and cumbersome.

o Use typedef (or using) and auto to clean up the ugly syntax of vector declaration.

• Use smart pointers such that you do not directly invoke dynamic memory allocation anywhere in your code.

o Use the C++14 syntax to avoid the keyword new.

o The only allowable use of raw pointers is if you need to implement an observer.

**High First classification**

• Add an additional two players to the game. Their names are Neil and Mike.

o If a player lands on an assessment that more than one player has completed, all those players who

have completed the assessment join in helping the player who has landed. The success is shared equally (rounded to the nearest whole number) between the player and their helpers.

• Implement the following extra rules about Mitigating Circumstances and deferring assessments, to be applied at the end of each player’s turn:

o If a player’s motivation has fallen below zero, they may look back on assessments they have already completed to extract motivation from them. To do this, the player must apply for Mitigating Circumstances and defer handing in one or more of their completed assessments until their motivation rises above zero.

▪ Assessments must be deferred in motivational cost order, starting with the lowest cost assessment first.

▪ When MCs are (automatically) granted, the assessment is deferred. The player regains the original motivational cost of the assessment, but loses the associated success.

▪ Note that a player cannot help another player with an assessment that they have deferred.

o If a player is unable to reach positive motivation by deferring all their assessments, they immediately quit the course and are out of the game.

o If a player’s motivation rises above the cost of their lowest valued deferred assessment, they must

complete that assessment. It will cost them motivation, but their success will be restored.

▪ Completing a deferred assessment cannot result in a player having negative motivation.

▪ Output appropriate messages when properties are mortgaged or redeemed, e.g.'<Player> defers <assessment> for <motivation amount>• Increase the number of rounds (set an upper limit of 500) to check that the game stops when either all students have dropped out or one of them graduates.

**Appendix**

Reading the " degrees.txt" file

For all of the scenarios the file can be read in a straight forward fashion. In order to make reading the file straightforward a code number is used. At the beginning of each line there is a number which identifies the type of square stored on that line. Use the code number in order to determine whether the data refers to a assessment, a Station, one of the special squares, etc. Code Meaning1 Assessment2 Welcome Week3 Extra-curricular Activity4 Bonus5 Bogus6 Plagiarism Hearing7 Accused of Plagiarism8 Skip Classes. The file will always be in the same format. You can assume that an Assessment has a name consisting of two words, followed by its motivational cost, then success score, then year, etc. Number generation (the Spinner) There is a code file labelled “random.cpp” This implements a function called “Random”. Random returns a randomly generated number in the range 1 to 10.Random number generators only generate a pseudo-random sequence of numbers. If the generator is seeded with the same number then an identical sequence of numbers is generated.• The seed for the random number generator is read from the file "seed.txt".• The use of a seed will allow me to check your work against a known number sequence.• I will supply examples of play using a particular seed a week or two before the deadline. Seed the generator with the given number and the play will be exactly the same each time. You can use the examples of play to test your program.

**Number generation (the Spinner)**

There is a code file labelled “random.cpp” This implements a function called “Random”. Random returns a randomly generated number in the range 1 to 10.Random number generators only generate a pseudo-random sequence of numbers. If the generator is seeded with the same number then an identical sequence of numbers is generated.

• The seed for the random number generator is read from the file "seed.txt".

• The use of a seed will allow me to check your work against a known number sequence.

• I will supply examples of play using a particular seed a week or two before the deadline. Seed the generator with the given number and the play will be exactly the same each time. You can use the examples of play to test your program.