TDDE18/726G77 - Exam

2018-01-12

Rules

- All code sent for assessment should compile and be well tested.
- Electronic devices are not allowed. Phones should be switched off and placed in a coat or bag.
- Outdoor clothes and bags should be placed in designated area.
- All contact between students are strictly prohibited during the exam.
- Books and notes may be reviewed by invigilators during the exam.
- Questions regarding specific assignments or regarding the exam in general should be asked through the communication client.
- System questions can be answered by assistant if you raise your hand.
- Assignments sent in after exam end will be disregarded.
- You can correct flaws and ask for new assessment until an assignment have grade “Pass” or “Fail”. An assignment can be assessed as “Fail” if no significant improvement took place since last attempt.
- Correctly compiling code, complete fulfillment of requirements, and use of good programming conventions and style are requirements for “Pass” grade on an assignment.

Aiding material	One C++-book One A4-page with any notes

Information

Grading guidelines - TDDE18

The exam consists of five assignments. Solutions that fulfill specification and follow good conventions are assessed “Pass”. Other solutions are assessed “Try again” or (rarely) “Fail”. Grading is based on the number of assignments with a passing grade you solve during the first four hours of the exam. See Table 2. *For grade 3 you always have the full exam time.*

Time	Solved assignments	Grade
17.00 $+B$	3	5
18.00 $+B$	4	5
18.00 $+B$	3	4
16.00 $+B$	2	4
19.00	2	3

Table 1: Grading, 5 assignments given

Grading guidelines - 726G77

The exam consists of five assignments. Solutions that fulfill specification and follow good conventions are assessed “Pass”. Other solutions are assessed “Try again” or (rarely) “Fail”. Grading is based on the number of assignments with a passing grade you solve during the first four hours of the exam. See Table 2. *For grade 3 you always have the full exam time.*

Time	Solved assignments	Grade
15.30 $+B$	2	VG
16.30 $+B$	3	VG
17.30 $+B$	4	VG
19.00	2	G

Table 2: Grading, 5 assignments given

Bonus time ($+B$)

Bonus time is valid only during the first regular exam after the course (January). Each met deadline in the course give 5 minutes bonus time for higher grade to a total of at most 40 minutes. This is shown as $+B$ in the table where valid.

Log on

Choose “Exam system” from the session menu in the lower left corner of the welcome screen. You should the log on using your normal LiU-ID. This will take you to the exam login. Select language (never mind the flag colors, look for the English flag pattern). Follow the instruction on screen until a one time password is requested. Have your LiU-card ready and show it to assistant or invigilator to get the password.

Desktop environment

Upon successful login you will enter the normal desktop environment (Mate-session in Linux Mint) with a green background. The communication client should start automatically. The start

menu is cleared of all but the tools required by the exam. Other tools may still be available from the command line. Not that the network is inaccessible, network applications may thus malfunction.
It is important that you leave the communication client running during the entire exam. We may send out public corrections and hints. Notify assistant if it does not start automatically within 5 minutes after login or after selecting the fish in the start menu.

Terminal commands

`e++17` is used to compile with “all“ warnings *as errors.*
`w++17` is used to compile with “all“ warnings. **Recommended.**
`g++17` is used to compile `without` warnings. `valgrind --tool=memcheck` is used to check for memory leaks.

C++ reference pages

During the exam, you will have access to `http://www.cppreference.com/` in the browser Chromium. Note that only this site is accessible, and that some features on the site may be blocked.

Given files

Any given files reside in the folder `given_files`. This folder is write protected, thus you need not worry about accidentally changing the given files. To modify a given file, you must first copy it to your home folder. You are expected to know how to do this, it is part of the course. The home folder is the folder you are in from start. You can always get back to it by typing `cd` in your terminal. The home folder is always named `/home/student_tilde` during the exam.

Log off

When your assignment and exam grade is satisfactory (and correct) in your communication client it is save to leave. If you run out of time you have to leave without knowing the result of your last attempt, contact the examiner by email after the exam to know the result.

Terminate all open programs, select “Logout” from the start menu, and confirm. Wait for a while, select “Finish exam” and confirm again. You can leave the computer once you see the normal welcome screen. Notify invigilator or assistant in case of problems.

Assignment 1: Dose Track

Keeping track of how much medicine you need to take could be a hassle. It would be even harder to keep track if you need to change the dosage based on what time in the day you need to take it. Some medicine is recommended to take more in the beginning and less for every hour. Your task is to create a program that can handle the following type of dosage:

1. constant dosage, where the total amount is divided by the amount of hours

Functional requirements

Your program will ask for total amount of medicine and total amount of hours that the dosage must be calculated for. Your calculations should only give the dosage in full integers (since we cannot split up the medicine pills). Higher dose is given first.

Non-functional requirements

1. Your solution must be readable
2. Your solution must be well structured and have good abstraction
3. Your solution must match the example below exactly

Examples (user input in bold font)

```
Welcome to dose track!

What is the total amount of medicine (in gram)? 22
Over how many hours do you want to take the medicine? 4
The dosage is: 6 6 5 5
```

Example 2 (user input in bold font)

```
Welcome to dose track!

What is the total amount of medicine (in gram)? 22
Over how many hours do you want to take the medicine? 6
The dosage is: 4 4 4 4 3 3
```

Assignment 2: Chess pieces

Introduction

Chess is a two-player strategy board game played on a chessboard, a checkered game board with 64 squares arranged in a 8x8 grid. The rank range from A to H horizontally and the file range from 1 to 8 vertically. Each player have 6 types of chess pieces: 1 king, 1 queen, two rooks, 2 knights, 2 bishops, and eight pawns. In this assignment we will focus only on three chess pieces:

- The pawn moves one square forward or two in the first move
- The rook can move any number of squares along a rank or file
- The queen can move any numbers of squares along a rank, file or diagonal

Functional requirements

Assume an empty board, we don't interact with other pieces. You are to create a class hierarchy that represent the chess pieces described. The chess pieces must know its current position on the board and when given a new position will either move there or throw an error if its not allowed to move there. You must also provide a minimal test program that shows polymorphic behaviour with your classes.

Non-functional requirements

1. You must create a class hierarchy for this assignment
2. The base class must be an abstract class
3. You must use polymorphism for the movement function

Tips for chess piece movement

1. Movement for a rook is valid when the difference in rank or file is 0.
2. For a queen the valid move is the same as rook, with the extension of diagonal movements. This is true when the difference in rank and file are the same.
3. A movement for pawn is valid if the file is unchanged and the difference in rank is 1. The first movement of a pawn also accepts a difference of 2 in rank.

Assignment 3: The Quiz

Introduction

Sam and his colleagues had a Christmas party quiz and he thought that it would be awesome to use the data from this quiz as an exam question.

Functional requirements

Create a program that allows the user to enter the file name of the quiz data as a command line argument. The program must print out the top 3 participant with the highest total score.

```
The format of the data file is in the following format (one per line):
Question Player Correct Score Score_no_bonus
```

(Question is question number, Player is player alias, Correct is whether the player answered correct or incorrect, Score is total score on that question, Score_no_bonus is total score without any bonus added).

Non-functional requirements

1. The name of the file must be specified on the command line
2. Every error during command line or file handling must be detected and handled by writing a clear error message to standard error. The program must then exit immediately
3. The output must match the example exactly

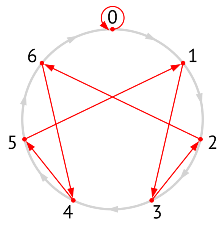
Example (user input in bold)

```
$ ./a.out
Error: No arguments given.
Usage: a.out FILE

$ ./a.out example
Error: No file named "example"
Usage: a.out FILE

$ ./a.out quiz_data.txt
1st place: August with 12160 total points
2nd place: Chris-Cross with 11944 total points
3rd place: Eric with 11623 total points
```

Assignment 4: Divisibility by 7 is a Walk on a Graph

Introduction

The first three primes have elementary techniques for checking divisibility. Two is a factor of any number ending in 0, 2, 4, 6 or 8, and 5 is a factor of numbers ending in 0 or 5. Any number with 3 as a factor has the curious feature that its digits add up to a number that's evenly divisible by 3, a property we'll take a closer look at later. Primes above 5 aren't so easy to check, but it turns out that neither are they as hard to check as doing the long division. Given the graph below the divisibility by 7 could be explained with the following algorithm:



1. Given a number **n**, break it into digits n_1, n_2, n_3, etc. where n_1 has the largest place value.
2. Put your finger on the above graph at zero.
3. Move your finger along the gray arrows n_1 times.
4. If there are any digits left to consider, move your finger along one red arrow, then repeat from Step 3 using the next digit. If there are no digits left, your finger is on the remainder of n when divided by 7.

As an example, let's see if 7 is a factor of 243. Starting at 0 at the top, go clockwise along the circle 2 steps to 2. Follow the red arrow to 6. Continue around the circle 4 steps to 3. Follow the red arrow to 2. Around the circle 3 steps to 5. The remainder when dividing 243 by 7 is 5, so 7 isn't a factor of 243.

Functional requirements

In the given file `Assignment4.cc` there is a test program that uses the `class Mod7`. Your task is to create the graph inside the `class Mod7` using the `struct Node` along with all the necessary functionality to make the test program work.

Assignment 5: Pig Latin - Must use Standard algorithms

N.B.

It is required to only use the standard library and standard algorithms to solve this exam assignment.

Introduction

"Ancay youyay eakspay igPay atinLay?" (Can you speak Pig Latin?) If you can't, here are the rules:

- After the 40th output character end the output with `...`
- If a word begins with a consonant, take all of the letters before the first vowel and move them to the end of the word, then add ay to the end of the word. Examples:
  - pig -> igpay
  - there -> erethay
- If a word begins with a vowel (a, e, i, o, u or y), simply add yay to the end of the word. For this, problem, y is always a vowel. Examples:
  - and -> andyay
  - ordinary -> ordinaryyay

Although there are many variants of Pig Latin, for this problem we will always use the rules described above.

Functional requirements

Your program will wait for input to translate. After each input word your program will print that word translated to Pig Latin. The program will terminate when you press Ctrl-D.

Non-functional requirements

1. You must show good knowledge about the standard library
2. You cannot use any (for/while/do-while)-loops in your solution. for_each is OK
3. The output must match the given example below

Examples (user input in bold font)

```
$. Enter the sentance to translate, one per line (finish by Ctrl-D):
i cant speak pig latin
iyay antcay eakspay igpay atinlay

$. Enter the sentence to translate, one per line (finish by Ctrl-D):
this is an exam day and
we wish you all good luck on the exam
isthay isyay anyay examyay ayday andyay ...
```