TDDE18/TDDD33/726G77 - Exam

2019-01-14

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.
- Students may leave no earlier than one hour after exam start.
- Fill in by invigilators designated list if you need to leave the room.
- 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/TDDD33

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	3	5
18.00	4	5
18.00	3	4
16.00	2	4
19.00	2	3

Table 1: Grading, 4 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
16.30	2	VG
17.30	3	VG
17.30	4	VG
19.00	2	G

Table 2: Grading, 5 assignments given

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 log in. 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 log in 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 log in 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: DNA

You want to rule the world, but to succeed with this task you need a group of gen-manipulated minions. Unfortunately it takes to long to try out all the possible DNA sequences.

You cooperate with your colleague, Professor Vim, and came up with these replacement rules:

```
A -> AT
T -> GC
C -> TA
```

You will begin your DNA sequence with "A" and for each iteration you will write out the new DNA sequence.

Repeat this procedure until all iterations are done. Every iteration must be written to the terminal.

Function requirements

1. The user will enter the number of iteration for the replacement rules.
2. The output must match the example.

Example (user input in bold)

```
How many iterations? 5
A
AT
ATGC
ATGCGTA
ATGCGTAGGCAT
ATGCGTAGGCATGGTAATGC
```

Assignment 2: Food party

You are about to host a dinner party for your friends. You have a selection of various foods such as pizzas, pizza rolls and salads. As with all people, your friends have various preferences when it comes to food and being a good host you want to accommodate all of your friends if possible. You have four types of friends; those who eat everything, those who love pizza but dislike pizza rolls, those who love salad, and those who are vegetarians.

Your assignment is to create class hierarchies of the various foods and guests at your dinner party. The following classes should be implemented:

Food abstract base class of all food items. This class contains a name and a pure virtual function `is_vegetarian`.

Pizza dervied class of `Food` which stores a `bool` that determines whether or not this is a vegetarian pizza. This class must override `is_vegetarian` such that it will return the `bool` variables value.

Salad Derived class of `Food` which overrides `is_vegetarian` such that it always returns `true`.

Guest base class of all guest types. Contains the name of the guest, a virtual function `prefer` which takes an object of type `Food` and returns `true` and a function `eat` which takes a `Food` object and prints a message whether or not the guest prefers that food. The message should have the following format: `<name of guest> eat <name of food>.` if the guest prefers the food and `<name of guest> does not want <name of food>.` if the guest does not prefer the food.

Salad_Lover derived class of `Guest` which overrides `prefer` such that it returns `true` only when the supplied `Food`-parameter is of type `Salad`.

Pizza_Lover derived class of `Guest` which overrides `prefer` such that it returns `true` if the `Food`-parameter is *exactly* of type `Pizza`, otherwise it should return `false`.

Vegetarian derived class of `Guest` which overrides `prefer` such that it returns `true` only when `is_vegetarian()` is `true` for the given `Food`-parameter.

There is a skeleton for a test program given in assignment2.cc.

Assignment 3: Stocks

Introduction

You will save money by taking the bicycle instead of the car. This money could be invested into stocks. For a couple of years, you and your friends have invested in stocks. And it is finally time for you to sell all your stocks to buy a electric bike. The problem is you don't know how much money you have saved.

Functional requirements

Create a program that allows the user to enter the file name of the stock data as a command line argument. The program must print out the amount of money each person have. The program will also print out the value of each stocks combined.

`The format of the data file is in the following format (one per line):`
`Name_Stock_Amount`

(Name is the persons first name, Stock is the stock, Amount is how much the stock is worth in SEK)

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 stock_data.txt
Ericsson 16000
Andersson 1998
Christoffer 999

Ericsson 16998
Saab 1999
```

Assignment 4: Sorting list

Introduction

Given a linked-list that is not sorted. Your job is to use something called "Selection sort" to get it sorted from least to most. "Selection sort" is a simple sorting algorithm that steps through the list to be sorted, select the highest number, removes it and insert that number to the beginning of the list. It will then continue to search for the next highest number, remove that number and insert it again in the beginning of the list. This will continue until the list is sorted.

Functional requirements

Sort a linked-list in an ascending order given a list in the given file `assignment4.cc`.

Non-functional requirements

1. The list must be sorted in an ascending order using selection sort.

Tips

- You can remember the position from which the list is not sorted.

Examples

```
Before sort ( 5 1 4 2 8 )
After sort ( 1 2 4 5 8 )
```

Assignment 5 - Debt

When multiple people take turns buying lunch to each other, it can be hard to keep track of all the debts. In this assignment, you shall create a program that allow the user enter any number of purchases in the format
"`<Payers initials> <Receivers initials> <price>`" and afterwards printout a summary of all the debt given in the program example below.

Functional requirements

- The program allow the user enter the debt in the standard input (cin) until the end of file (ctrl-d).
- The printout shall be sorted on amount owed, from most lend to most in debt.
- The format on the printout must match the program example below. A positive value means that someone is lending money (doesn't matter to whom) and a negative value indicates that this person should buy lunch the next time.

Non-functional requirements

- You must use an appropriate STL-container to store the information about the debts.
- You must sort the STL-container accordingly to the functional requirements.

Tips

- The sum of all values in the table must be 0.
- You don't need to know who is owing who money. It is only needed to know the total debt/owing for each person.
- You might need to use multiple STL-container, one to store and one to sort.

Program example

```
AE EM 102
EM BD 23.1
CK EM 15
FF CK 17.2
AE CK 9.95
DH FF 35.2
DH EM 65.8
FF CK 18
AE EM 91.55
<CTRL-d>
Name  Balance
-------------
AE     203.50
DH     101.00
FF       0.00
CK     -30.15
BD     -32.10
EM    -251.25
```