Practical-Exam-03: OOP and Recursion

Due 13 Sep by 12:00 **Points** 100 **Available** after 13 Sep at 10:15

Introduction

This session ends 12 pm

Make sure to complete your submission in time.

Submit to SVN & Websubmission (for every problem solved)

Submission

```
svn mkdir -m "first assessment commit" --parents https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-07/practical-exam-03

Files heading:

//=================================
// Foundations of Computer Science
// Student: you name
// id: your id
// Semester:
// Year:
// Practical Exam Number:
//==================================
```

Assessment

- **This is a practical exam - your work must be entirely your own.**
- Marks for this practical exam contribute 2 marks for the final mark of this course.
- Style marks will be awarded later by your workshop supervisor (50%)
- Functionality marks will be awarded by WebSubmission marker (50%).
- **Due date: the end of this section (12 pm).**
- **Do Not Forget** To Submit on **WebSubmission**
- **Late penalties:** Only work submitted during your enrolled practical session from a Linux system in the practical lab will be marked.

```
(1) only codes that can compile will be marked;
(2) only codes that are in the suggested directory will be marked;
(3) only codes submitted to SVN and WebSubmission before the deadline will be marked;
(4) only codes containing your signature on the top of the file will be marked by tutors;
(5) you will have your markers decreased in 5 points if *.class file present in your folders;

To acquire full marks (1) all your functionalities must work perfectly,
(2) your code has to be well and proportionally commented, and
```

```
(3) your code must follow correct indentation (4 spaces for every new indentation level)
and (4) you have to use all the content from latest lectures.
```

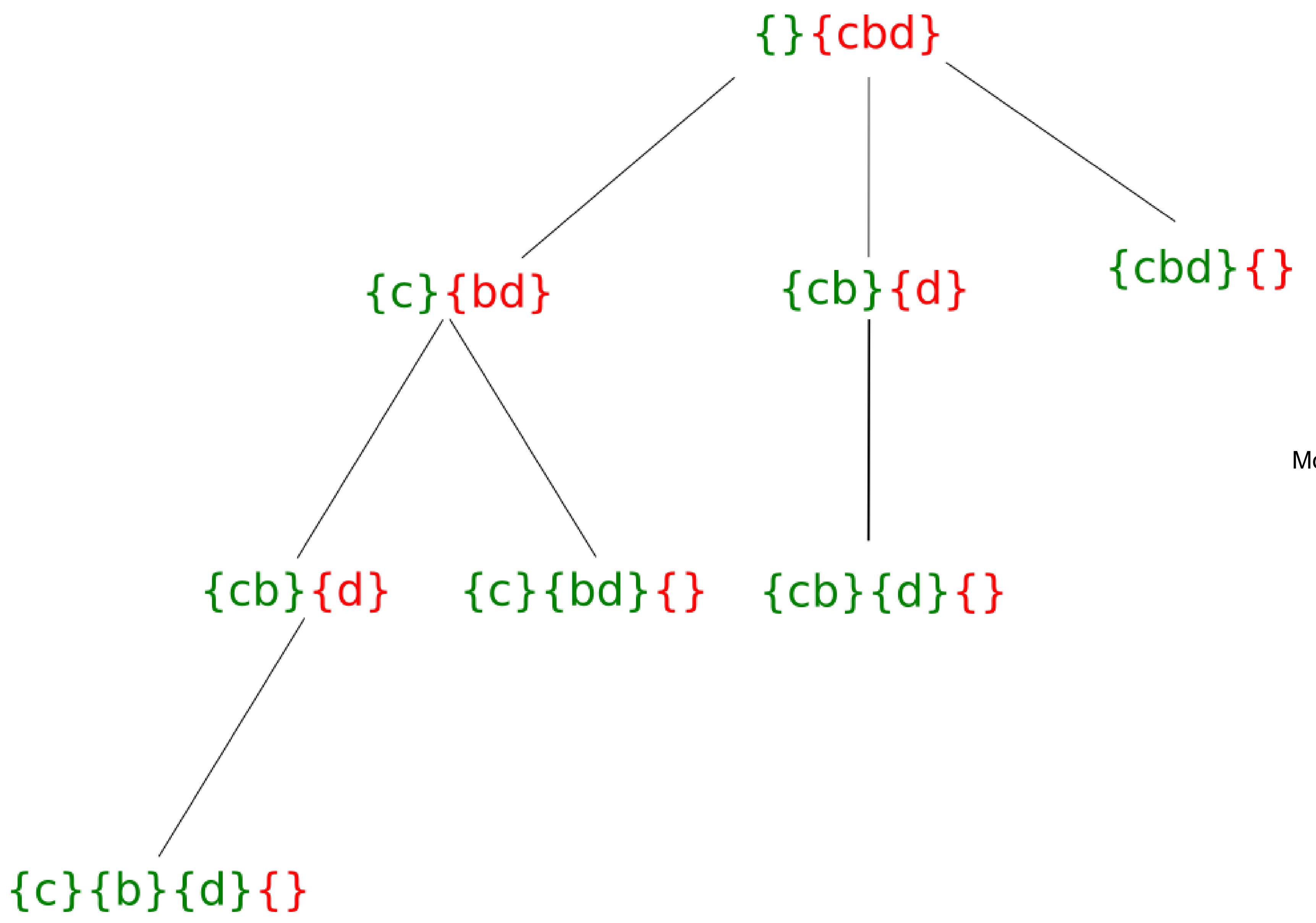
Practical Exam 03

Criteria 01: The problems MUST extend the following Abstract class:

```
abstract class ProblemAbstract {
    public abstract void solve(String word);
}
```

Problem 01

In this problem you are required to implement a solution for `String` decomposition. Given a string, your code should be able to decompose in ALL the possible non-overlapping combinations. For example, the String `cbd` has 4 possible decompositions, which are `{cbd}, {cb}{d}, {c}{bd}, {c}{b}{d}` . In the following diagram we depicted a basic recursive solution for this problem. The diagram show a recursive tree for the string `cbd`, where every recursive call is parameterized by a string in process, and the unprocessed part, green and red respectively. Please note, that your algorithm should follow the same order of solution as the example -- otherwise, the websubmission might not grant you the marks.



Moreover, we provide an example solution for the problem `CBD` and `ABCD`:

```
*** note this is example only, you are not required to implement
 a user-interface to acquire information from the user. However, your
 code should display the results as formated before, using { } ***

For the string CBD:
{CBD}
{CB}{D}
{C}{BD}
{C}{B}{D}

For the string ABCD:
{ABCD}
{ABC}{D}
{AB}{CD}
{A}{BCD}
{A}{BC}{D}
{A}{B}{CD}
{A}{B}{C}{D}


Constraints:
(1) save as .../problem-01/Problem.java;
(2) In Problem.java extend ProblemAbstract.java;
(3) expected output should contain { } and blank line at the end;
```

```
(4) Recursive solution will be fully awarded in terms of functional marks;
Other solutions will be penalized during tutor marking;

----------------
Example output:
{UoA}
{Uo}{A}
{U}{oA}
{U}{o}{A}

----------------
```

Problem 02

In this problem, you are required to define a solution for a palindrome `String`, where your code MUST indicate whether a `String` is a palindrome or not.

In general terms, a word is considered to be a palindrome if the characters remain the same after reversing the order of this word. For instance, the `Strings`: **ABCBA** and **GLENELG**, and a few more examples:

```
reverse(GLENELG) == GLENELG, therefore this String is a palindrome.

reverse(ABCCBA) == ABCCBA, therefore this String is a palindrome.

However,

reverse(HOUSE) != HOUSE, then this String is not a palindrome.


Constraints:

(1) save as .../problem-02/Problem.java;
(2) In Problem.java extend ProblemAbstract.java;
(3) Recursive solution will be fully awarded in terms of functional marks;
Other solutions will be penalized during tutor marking;

----------------
Example output:
QUEEN: non-palindrome

----------------
```

Basic Marking Scheme II

Criteria	Ratings	Pts
Compilation In order to achieve full marks - your code must (1) compile, (2) run, (3) no upload *.class;		15.0 pts
Basic Functionality Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem. Please note: 75% are being marked by WebSubmission; 25% being marked by tutors.		45.0 pts
Code Formatting Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read.		40.0 pts
		Total points: 100.0