

# Week 12 Practical Exam R/AA

---

**Due** 3 Nov by 10:00    **Points** 100    **Available** 3 Nov at 9:00 - 3 Nov at 11:00 about 2 hours

---

This assignment was locked 3 Nov at 11:00.

## Weighting and Due Date

- Marks for this practical exam contribute 5% of the overall course mark.
- All marks will be awarded automatically by the web submission system.
- **Due date: 10am Tuesday 3 November 2020.**
- **Late penalties:** late submissions will receive a mark of 0.
- **Core Body of Knowledge (CBOK) Areas:** abstraction, design, hardware and software, data and information, and programming.

## Replacement / Additional Assessment

This exam is a repeat of the [Week 12 Practical Exam](#) but the grammar to be parsed will be different. There will be no more opportunities for a replacement or additional assessment. **Do not attempt this exam if you are happy with your original mark.**

**Replacement Practical Exam:** If you take this exam as a replacement exam, your mark for this exam will replace your mark for the original exam, even if it is lower. However, your final mark will not be reduced below 50%.

**Note:** A replacement practical exam may be offered if appropriate written evidence is provided.

**Additional Assessment:** If you take this exam as an additional assessment, your mark for this exam will replace your mark for the original exam, if it is higher. Your mark cannot go down. However, your final mark will not be increased above 50%.

**Note:** All students with a mark less than 50% can take this exam as an additional assessment.

## Exam Setup

**Note:** this exam assumes that you have already created directories for every assignment, workshop, project, and exam in your svn repository, as described on the [Startup Files for Workshops and Assignments](#) page.

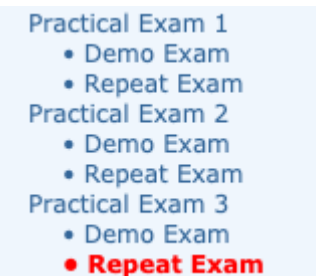
1. If required, checkout a working copy of the **exam3-repeat** directory from your svn repository.
2. Change directory to the working copy of the **exam3-repeat** directory.
3. Copy the newest zip file attached below into the updates sub-directory and add it to svn.
4. Run the following command to place the exam's startup files in the correct locations:

```
% make install
```

5. svn add the **parser.cpp** file to your svn repository and perform an svn commit.

```
svn add parser.cpp  
svn commit -m exam3r
```

6. Make a submission to the [web submission system \(https://cs.adelaide.edu.au/services/websubmission\)](https://cs.adelaide.edu.au/services/websubmission) assignment named: **Practical Exam 3 / Repeat Exam**



Practical Exam 1  
• Demo Exam  
• Repeat Exam  
Practical Exam 2  
• Demo Exam  
• Repeat Exam  
Practical Exam 3  
• Demo Exam  
• **Repeat Exam**

## Exam Description

You will be required to complete a C++ implementation of a recursive descent parser for a simple language by editing the file **parser.cpp**. The parser constructs an abstract syntax tree for the program being parsed in a very similar way to workshop 10.

At the start of the parser.cpp file there is a copy of the grammar that you must parse. This has been written in terms of tokens and token groups using their tk\_\* and tg\_\* names respectively. This is followed a list of the provided token groups.

The **parser.cpp** file already includes the code required to create and delete symbol tables. You just need to add the appropriate calls to the **declare\_variable()** and **lookup\_variable()** functions.

The web submission test script will run a set of tests that you have been given in addition to a further set of secret tests. If you correctly implement the grammar using an appropriate combination of ***have()***, ***mustbe()*** and ***did\_not\_find()*** functions, you should pass all the tests.

## Testing

The startup files have the same structure as used in the workshops and programming assignments. If you wish to test your implementations without making a web submission, you can simply run the command:

```
% make
```

```
Make a web submission after completing each part of the exam!
```

```
student parser against Pxml files
```

	Test Input	parser	Expected Test Output	Test Result
Checking	"cat tests/some.test	./parser	./bin/diffc - tests/some.test.Pxml	" - test passed

To see where your implementation is failing you can use the commands shown between " " in the output of make to see the relevant error messages.

**Note:** You have only been given a small set of test examples. If your program passes these tests it does not mean that it is correct. It only means that it appears to work with the example test cases on your computer. If the examples pass on your computer but fail on the web submission system, you have a bug in your program that you need to fix. Remember that the order in which function arguments and operands in expressions are evaluated can differ from one C++ compiler to the next. If your program fails in this way it is incorrect.

## Marks

Your mark will be the percentage of tests that a submission passes.

## Frequent Web Submissions

Remember to make frequent submissions to the web submission system

- late submissions receive a mark of 0, so

- make a web submission as you complete each parsing function.

## Startup Files

The following zip file must be placed in the updates sub-directory and added to svn. When make is run, the zip file in the updates directory is used to update the startup files. Any files you are required to edit will not be updated but, a copy of the latest version of those files will be placed in the sub-directory originals.

- [exam3r-20200919-144414.zip](#)