UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS110 - Program Design: Introduction
Practical 7

1 Introduction

Deadline: 9th of October, 18:00

1.1 Objectives and Outcomes

The objective of this practical is to test your understanding of the programming concepts covered in the theory classes. In particular, this practical will test your understanding of templates in terms of classes, extending templates from functions to entire classes.

1.2 Submission

All submissions are to be made to the **assignments.cs.up.ac.za** page under the COS 110 page, and for the correct practical slot. Submit your code to Fitchfork before the closing time. Students are **strongly advised** to submit well before the deadline as **no late submissions will be accepted**.

1.3 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone elses work without consent, copying a friends work (even with consent) and copying textual material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to **http://www.ais.up.ac.za/plagiarism/index.htm** (from the main page of the University of Pretoria site, follow the *Library* quick link, and then click the *Plagiarism* link). If you have questions regarding this, please ask one of the lecturers, to avoid any misunderstanding.

1.4 Implementation Guidelines

Follow the specifications of the practical precisely. For each practical, you will be required to create your own makefile so pay attention to the names of the files you will be asked to create. If the practical requires you to submit additional files of your own, follow the file structure and format exactly. Incorrect submissions will use up your uploads and no extensions will be given. In terms of C++, unless otherwise stated, the usage

of C++11 or additional libraries outside of those indicated in the practical, will not be allowed. Some of the appropriate files that you submit will be overwritten during marking to ensure compliance to these requirements. If the specification makes use of text files, for providing input information, be sure to include blank text files with the specified names. Your submission archives should be in either .zip or .tar format.

1.5 Mark Distribution

Activity	**Mark**
Bin	10
Train	12
Total	**22**

2 Practical

2.1 Template Classes

A template class is very similar to a template function. By applying the template framework to an entire class it enables the creation of a generic class that can generically implement a number of types at run-time. In this way, template classes extend the ability of code to be generic and generalisable for a number of situations. This practical tests this concept by the creation of template classes that should be able to be used more generically than the classes that have been used thus far.

Additionally, you will be not be provided with mains. You must create your own main to test that your code works and include it in the submission which will be overwritten during marking.

2.2 Task 1

This will consist of implementing two classes: **train** and **bin**.

2.2.1 Bin

The class is described according to the simple UML diagram below:

```
bin <T>
-item:T*
--------------------
+bin(t:T)
+~bin()
+getData() const:T
```

The class has the following variables:

- item: A generic variable that is used to represent a variety of potential cargoes.

The class has the following methods:

- bin(t:T): The constructor for the class. It receives a generic argument.
- ~bin(): The destructor for the class. It deallocates the memory allocated by the class.
- getData(): This returns the value of the item.

2.2.2 Train

The class is described according to the simple UML diagram below:

```
train<T>
-storage: bin<T> **
-bins:int
-------------------
+train(numBins:int)
+~train()
+addBin(item:T):int
+addBin(index:int,item:T):int
+removeBin(index:int):int
+removeBin(index:int,type:T):int
+printStorage():void
```

The class has the following variables:

- storage: A dynamic variable that stores a number of bin objects. It should be able to store a variety of different types of bins but only one type of bin per train instance.
- bins: The maximum number of bins.

The class has the following methods:

- train(numBins:int): The constructor for the class. It initialises all of the class variables.
- ~train: The destructor for the class. It deallocates the memory assigned to the class. Bins should be deleted from the first index.
- addBin(item:T): This function adds a new bin at the first available index. If there are no indices available, it should return -1. Otherwise it should return the index of where the new bin was added.
- addBin(index:int,item:T): This function adds a new bin at the given index. If there is a bin located there, the function returns -1. Otherwise it should return the index of where it was added.

- removeBin(index:int): This removes a bin located at the provided index. If the index is out of bounds or the bin is already removed, then it should return -1. Otherwise it should return the index of where it was removed. The array should not be rearranged by the deletes.

- removeBin(index:int,type:T): This an overload of the removeBin function that behaves in the same way as the other function, with an additional condition, namely that it only performs the remove of the bin if the provided type variable (that is passed in) is the same as the data type of the bin being removed. That is, the variable type will be of data type T which is generic. The element should only be removed if the data type of the bin is the same as the data type of the type variable. The array should not be rearranged by the deletes.

- printStorage: This prints out the information contained within the class. It should start from the first index where a bin is located until all of the bins have been printed. The format is as follows:

  ```
  Bin 1: 142.2
  Bin 2: 120
  Bin 3: NA
  Bin 4: one hundred and ten
  ```

  If there is no bin located at the given index, it should print NA but otherwise print out the data contained in the bin with new lines at the end of each line.

You will be allowed to use the following libraries:

- train: iostream, typeinfo

The use of namespace std is only available in the main. You should compile all of your classes as you have previously done as individual classes then linked together into the main. Your main will need to include a train.cpp alongside the train.h and you should include a bin.cpp in the train.cpp.

You will have a maximum of 10 uploads for this task. Your submission must contain **train.h, train.cpp, bin.cpp, bin.h, main.cpp** and a makefile.