# University of Wollongong

**School of Computing and Information Systems**

## CSCI251/851 Advanced Programming Autumn 2017

## Assignment 5 (Due: 11.59pm Week 12, Tuesday 23 May) 8 marks

## Aim:

Gain experience developing programs using STL containers and inheritance.

**On completion you should know how to:**

- Write C++ code using STL container class objects.
- Use inheritance to facilitate programming with objects.
- Gain experience writing and debugging O-O programs incrementally.

## Prerequisites:

This assignment involves writing a program to handle inmates of a new prison. Information on the inmates is contained in the file "crims.txt". The criminal records are to be stored in an STL set to enable easy addition of new prisoners and the clearing of those inmates who have completed their sentence and have been paroled. The files: prison.h, prison.cpp, crim.h, crim.cpp and main.cpp have a partially completed object oriented implementation of the prisoner record system. Your task is to complete the implementation by performing the following steps.

**Step 1 (3 marks)**

Complete the function definitions of class Criminal and class Prison so that the data in "crims.txt" can be read into the set container and displayed on the screen in the order of the prisoners' names. Please note that the set is a set of pointers to Criminals. So when you read each record from "crims.txt" you will have to allocate a new instance class Criminal with the new operator and insert the pointer into the set. E.g.

```
Criminal* C = new Criminal(FirstName, FamilyName, Crime, Months, i);
Crims.insert(C);
```

ReadCrimFile() (in "prision.cpp") should assign a cell number sequentially to each prisoner as they are read. For now, ignore the extra name field on MURDERERS and the number field on ROBBERS (see "crims.txt"). Once you have ReadCrimFile() implemented, implement DisplayPrisioners() and the destructor of class Prison. The DisplayPrisoners() should iterate the Crims set and call the Print() function of class Criminal. The destructor should iterate the Crims set and delete each pointer in the set. Also, complete the class Criminal constructors, operator<, and Print() in "crim.cpp". The default destructor should initialise the Months to zero and the CellNo to -1. Operator< returns true if *this name is less than (before) the other passed name. Print should display records on the screen 10 at a time and shown by the example output below...

```
. . . .
Cell 71 AMSDEN, Terry [273 months for MURDER]
Cell 55 ASHCRAFT, Frances [90 months for ROBERY]
Cell 27 BARFIELD, Cristina [50 months for OTHER]
Display more records (y/n): n
```

**Step 2 (3 marks)**
To handle the extra information on robbers and murderers (in "crims.txt") declare two derived classes in "criminal.h". Put these class declarations below class Criminal. e.g.

```
class Robber: public Criminal {
  public:
    . . .
  private:
      int AmountStolen;
};

class Murderer: public Criminal {
  public:
      . . .
  private:
      string VictimsFirstName,VictimsFamilyName;
};
```

To enable the derived classes to inherit and access the private data members in class Criminal change "private:" to "protected:" in the class declaration of class Criminal. Then add a standard constructor to the derived classes (above). These should invoke the standard constructor of the Criminal base class and set the extra data member(s) in the derived classes. E.g.

```
Robber::Robber(char *FName,char *LName,char *Crm,int Mths,int
               Amnt,int Cell) : Criminal(FName,LName,Crm,Mths,Cell) {
     AmountStolen = Amnt;
}
```

Now overload the Print() function of class Criminal by making its declaration in class Criminal virtual:

```
virtual void Print();
```

and adding a public void Print() function to class Robber and Murderer which displays robbers and murderers differently. E.g.:

```
Cell 80 ENTWISLE, Winston [393 months for murdering SALKELD, Benny]
Cell  2 COMYN, Nigel [46 months for stealing $59000]
```

Alter ReadCrimFile() in "prison.cpp" so that after the Months field is read, test the crime and:

*if crime is "ROBBERY" read the amount stolen, insert a new class Robber to Crims;*
*if crime is "MURDER" read the victim's name, insert a new class Murderer to Crims;*
*else insert a new class Criminal to Crims;*

Now run your program and check that the output is appropriate for the crime committed.

**Step 3 (2 marks)**
Add 2 more options to the main menu:

```
(3) Reduce sentence
(4) Check for parole
```

ReduceSentence( ) should reduce the remaining months of all the inmates by 6 months (or less if zero is reached) and ask the user to release prisoners if their release date has been reached (i.e. months remaining has reached zero). If the user replies with a 'y' the record is removed from the Crims set. If 'n' is entered, the months remaining is set to 6 mths. E.g.:

```
Command> 4
2 prisoners have reached their release date.
Cell 64 PEDLEY, Laurance [0 months for TRAFFIC]
Cell 35 WILMER, Alberto [0 months for robbing $88200]
Do you want to release Laurance PEDLEY (y/n)? : y
 Laurance PEDLEY has been released.
Do you want to release Alberto WILMER (y/n)? : n
Alberto WILMER's sentence has been increased to 6 months.
```

CheckForParole() should display all prisoners who are ready for parole and ask the user to release prisoners if their parole date has been reached. Murderers become eligible for parole when their months remaining is 6 months or less, Robbers become eligible for parole when their months remaining is 9 months or less, and all others become eligible for parole when the months remaining is 12 months or less, If the user replies with a 'y' the record is removed from the Crims set. If 'n' is entered no change is made. E.g.:

```
Command> 4
2 prisoners have reached their parole date.
Cell  4 BEGLEY, Stacey [11 months for OTHER]
Cell 28 BERRIDGE, Allyn [4 months for murdering BUCKLIN, Kurt]
Do you want to parole Stacey BEGLEY(y/n)? : y
Stacey BEGLEY has been paroled.
Do you want to parole Allyn BERRIDGE (y/n)? : n
Allyn BERRIDGE parole has been refused.
```

Note: ReduceSentence() should be implemented with just a base class function in class Criminal. CheckForParole() should be implemented with a virtual base class function in class Criminal and overwritten in the derived classes (Robber and Murderer).

## Step 8 (for CSCI851 students. 1 mark deduction if not done.)

Implement a menu option '5' for adding a new prisoner to the Crims set. E.g.:

```
Command> 5
First name: Evil
Last name: Kenevil
Crime: traffic
Sentence (mths): 12
Cell 101 KENEVIL, EVIL [12 months for TRAFFIC]
... has been added the prison.
```

Note: Make sure you also ask for the amount stolen for robbers and the victim's first name and last name for murderers.

## Submit:

Submit your files (and the output produced (copy & paste) when the text in input.txt is entered in response to the user prompts) using the submit facility on UNIX as shown below:

**$ submit -u login -c CSCI251 –a5 main.cpp prison.h prison.cpp crim.h crim.cpp output.txt**

**where 'login' is your UNIX login ID**
Note: CSCI851 should also submit to –c CSCI251.

**You must demonstrate your program in the lab during your week 12 or 13 lab class. Failure to demo your assignment by 18:30 Tuesday week-13 will result is the work being marked by a tutor and a 1 mark deduction being given for lateness.**

Deductions will be made for untidy work or for failing to comply with the submission instructions. Requests for alternative submission arrangements will only be considered before the due date. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the marks awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than three days late.