Assignment 2 - part 2

Deadline	Anytime before Tuesday, 15th May 2018, midnight
Evaluation	10 marks –which is 5% of your final grade
Late Submission	5% per hour (or fraction of hour) it is late
Team	The assignment can be done individually or in pairs (of at most 2 students)
Purpose	Practice with designing and using template classes, memory allocation.

Problem to solve

You will design a template class. The name of the template class is `MyVector`. The class `MyVector` is a simplify version of vector that models a dynamic array. You may assume that the type `T` supports the assignment operator `operator=()`. Write both the class prototypes, implementations and all helper function in a file called `a2p2.h`. Do not include the `main` function in this file.

Hand-in: Submit `a2p2.h` electronically using the submission form on STREAM. DO NOT include the main function in this file.

If you have any questions about this assignment, please ask the lecturer before the assignment is due.

Problem statement

You are designing a `MyVector` class with the following specifications:

1. **Behaviours:**
    a. `MyVector` can store any number of items, and automatically expands its capacity if more memory is needed.
    b. Items can be assigned and accessed via the `[]` operator.
    c. It keeps track of its capacity and the actual number of items. The capacity never shrinks, unless when the method `clear()` is called.
    d. The capacity is only expanded when the number of items is going to exceed the current capacity.
    e. The capacity is expanded in the following manner:
        i. if the current capacity is 0, expand it to 1
        ii. otherwise, double its capacity. e.g. if you incrementally insert 10 items into a default constructed `MyVector` object, its capacity will change as following: 0, 1, 2, 4, 8, 16 while its size will change incrementally (0, 1, 2, 3, … 10)

2. **Public methods:**
    a. Default constructor: sets capacity and size to zero, does not allocate memory.
    b. Constructor with one `int initialCapacity` argument: sets capacity to the value of this argument and size to zero. Allocates memory accordingly. Throws an `invalid_argument` exception if `initialCapacity` is negative.
    c. Copy constructor
    d. Move constructor
    e. Copy assignment operator
    f. Move assignment operator
    g. Destructor
    h. Output operator `operator<<`: must be non-friend.
    Access operator `operator[](int index)`. Throws an `invalid_argument` exception if `index` is out of bound
    In this case, out of bound means `index` is either negative or `>=` the current size. E.g. if you have a vector with 5 elements {1, 3, 10, 4, 5}, the accepted range of `index` is 0-4 (if `index == 5` it should throw an exception)
    i. `int getCapacity`: returns the current capacity.
    j. `int getSize`: returns the current number of items.
    k. `void clear`: resets `MyVector` object to the initial condition (capacity =0; size=0; all allocated memory deleted)
    l. `void pushBack(const T& item)`: Adds the item to the end of `MyVector` object
    m. `void insert(const T& item, int position)`: Inserts an item at given position. Throws an `invalid_argument` exception if `position` is out of bound
    In this case, out of bound means `position` is either negative or more than the current size. E.g. if you have a vector with 5 elements {1, 3, 10, 4, 5}, the accepted range of `position` is 0-5 (if `position == 5` that is the same as calling `pushBack`)
    n. `T remove(int index)`: removes item at given index and return this item to the caller. Throws an `invalid_argument` exception if `position` is out of bound. Does not change capacity.
    Out of bound in this case is the same as in the case of `operator[](int index)`.

    o. `bool isEmpty`: Returns true if the `MyVector` instance has no item and false otherwise

3. **One public function (not member of `MyVector`):**
    a. `void printStudentInfo`: displays your names/IDs on screen

Test your class:

Download `main.cpp` from Stream. That file is a simplified version of the actual main function that will be used to evaluate your program. Make sure you can compile that file with your class.

Sample output

Your new classes must make the given `main` function compile and print out the following output EXACTLY (Replace Jane Doe with your name(s) and 1234567 with your ID(s)):

```
**************************************
* Assignment 2 Part 2 *
* Jane Doe *
* 1234567 *
**************************************
Should print out 10 11 13
10 11 13
Should throw an exception
Exception thrown: Index out of bound.
Should print out 10 11 12 13
10 11 12 13
Should print out 10
10
Should print out 11 12 13
11 12 13
Should print out 11 120 13
11 120 13
Should print out nothing

Should print out 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
Should print out 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
Should print out nothing

Should print out 0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

Your program will be compiled, run and evaluated by a machine. So besides your name(s) and ID(s), the output must be EXACTLY the same, character to character.

Requirements

What you must and must not do

1. Your program MUST be named **`a2p2.h`**. If you use a different name, you will get zero.
2. File **`a2p2.h`** must not contain any extra class or public function. Your **`MyVector`** class may have extra PRIVATE functions if you want to, but it MUST NOT have any extra PUBLIC function.
3. You MUST NOT use any built-in container other than raw array. If you implement your class using **`vector, list, queue, deque,`** etc. you will get zero.
4. Your program MUST NOT require any user interaction.
5. You MUST NOT include any non-C++ standard library header files. E.g. **`window.h`** is a Windows-specific header and you MUST NOT include it.
6. You MUST NOT use the **`exit()`** function.
7. You MUST follow camelCase style.
8. Your class SHOULD be organised as followed:
    1. Header (Your name(s), ID (s), short description for the program, etc)
    2. All included files/libraries

    3. Class declaration
    4. Class definitions

9. All functions and classes MUST be properly documented following Doxygen format. See http://www.yolinux.com/TUTORIALS/LinuxTutorialC++CodingStyle.html for detail. You will also learn how to document your code in the tutorials
10. You MUST NOT use advanced features not covered by the course by the due date

Miscellaneous

1. When working in pair, send one solution file per team.
2. The assignment will be discussed on Wednesday lecture before the assignment is due and solutions will be discussed on Monday lecture after the due time.
3. Marks will be allocated for: correctness, completeness, consistent coding style, sensible structure (simple and clear solution, good use of helper functions), good understanding of class inheritance, good documentation.
4. Using goto, non-constant global variables or C-like I/O constructs (i.e printf fprintf, scanf, FILE*,etc) is not allowed and it will be penalised. Only const global variables are allowed.
5. Programs that do not run or do not compile in the (Albany) labs, using gcc(SciTe), get 0 marks.
6. The program must be your own work. Please be aware that you might be asked to explain to your lecturer how your program works. If you cannot explain it, then it is not yours and you will get 0 marks for that assignment. Attributing someone else's work as your own is plagiarism, and it is a violation of Massey University policy. We might file an official complaint against any student who we believe has committed plagiarism.
7. Suspicious similar solutions will all get 0 marks