| | | |
|---|---|---|
| **Name:**<br>*(as it would appear on official course roster)* | | |
| **Umail address:** | **@umail.ucsb.edu** | **section 1pm, 2pm, 3pm, 4pm** |
| **Optional: name you wish to be called if different from name above.** | | |
| **Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"** | | |

1

# h01: Templates and the STL

h01

CS32 W19

| ready? | assigned | due | points |
|---|---|---|---|
| true | Mon 01/07 09:30AM | Mon 01/14 09:30AM | 50 |

*You may collaborate on this homework with AT MOST one person, an optional "homework buddy".*

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE,
OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments;
in place of that, we drop the lowest scores (if you have zeros, those are the lowest scores.)

---

**Reading: Templates and the STL, PS 8.3, 17, 18**

1. (10 pts) Fill in the information in the header. The following are required to get the 10 "participation" points.
   - Filling in your name and umail address.

   Also: For paper submission PLEASE submit on ONE SHEET OF PAPER, double-sided if at all possible. If you must submit on two printed sheets write name on BOTH sheets and no staples, paperclips, or folded corners.

2. Suppose we have a C++ class called `Student`. There are many ways one can create a collection `Student` objects. Among these: one can use a normal C++ array or an STL `vector`, and one could create a collection of objects, or of pointers to objects (presumably allocated on the heap.)
   a. (2 pts) Write a line of C++ code that declares an array `a` that can hold 10 objects of type `Student`.

   b. (2 pts) Write a line of C++ code that declares an array `b` that can hold 10 pointers to objects of type `Student`.

   c. (2 pts) Write a line of C++ code that declares an STL vector `c` that can hold 10 objects of type `Student`.

   d. (2 pts) Write a line of C++ code that declares an STL vector `d` that can hold 10 pointers to objects of type `Student`.

3. (20 pts) In the previous problem, you were asked to write four different declarations, (a), (b), (c) and (d). Each of these has "pros" and "cons". There are also differences among them that we might describe as "neutral", for whether they are a pro/con depends on the context. For the "pros" and "cons" below, please indicate which of the letters above (a, b, c, d) the statement applies to. Note that in some cases, you may have to indicate more than one letter. Circle the letters to which the statement applies. If you mess up, cross out all the letters and list the letters in the space to the right.

| Hero or zero? | Choices |
|---|---|
| Con: Additional `#include` directives and `using` statements are required to use this approach. | a b c d |
| Pro: The item declared can be expanded beyond 10 items if needed. | a b c d |
| Pro: You can declare this item without actually creating any Student objects. | a b c d |
| Con: The class MUST have a default constructor, or this declaration is not permitted. | a b c d |
| Pro: All space for the item and the Students it contains are co-located in memory—on the stack if the item is declared as a local | a b c d |

Grading: for each answer, deduct one point if an item should be included and is not, or is included and should not be.

4. (3 pts) In your own words, what is the "problem" for which "templates for functions" is a solution? I'm looking for a **brief** description—a single sentence, or at most 2-3 sentences that get to the *central point*, a description of the *problem*, not a detailed description of everything you know about templates, and certainly *not* a sentence copied, word-for-word, from either textbook.

# 2

# h01

## CS32 W19

5. (3 pts) The C++ syntax for function templates includes this `template <class Item>`. The name `Item` in the example above is preceeded by the keyword `class`. One might infer from that that `Item` should be the name of some `class`, e.g. `class Student`, or `class Roster`, etc. But that is not a correct assumption. Explain why. **(BRIEFLY!)**

6. Let's say you have a function **`int calc(int`** `operand1,` **`int`** `operand2,` **`char`** `op)` where `op` is either '+' or '-'. Instead of overloading this function for each numerical datatype in C++ (e.g. also defining it for `float`, `double`, etc., you want to create a generic function that is far more flexible. In the space below, define `calc` as a template function. If op is neither '+' nor '-', print an error message on `cerr` and call the system function `exit(1);` (you may assume that `#include<cstdlib>` has already been done.)

   Note that once we've covered exceptions, that would be a better approach than the `cerr/exit` technique.

   Grading:
   - (2 pts) for correct template function header
   - (2 pts) for correctly handling the error
   - (2 pts) for function body being correct for non-error case.