"Exam 1"

Wednesday, February 13, 2013

- Do not open this exam until we instruct you to do so.
- This is a closed book, closed notes, closed electronics, closed neighbor, pencil and paper exam.
- This exam is designed to be completed in 50 minutes. However, you may take 120 minutes to work on the questions. You must immediately turn in the exam when we call for it.
- Write answers in the spaces provided. Indicate your final answer clearly. Cleanly erase or cross out any work you do not want graded. Show your work and explain your reasoning where appropriate; this will help to earn partial credit.
- Backpacks must be left at the front of the room.
- Electronic devices of any kind, including cell phones and calculators, must be completely turned off and stowed in your backpack.
- The exam must be written with a pencil.

1. Tell what is output by the following program. Be careful and precise about spacing and newlines.

```
void mystery(int a, string & word)
{
  word[a] = word[a] - 'A' + 'a';
  cout << word << a;
  a = a - 1;
}


int main()
{
  string s = "DIME";
  int a = 3;
  for (size_t b = 0; b < 3; b = b + 1) {
    mystery(a, s);
    a = a - 1;
  }
  cout << endl << a;
  return 0;
}
```

2. Suppose we defined a class to represent a *bag* of integers, some of which might be duplicated. Of the methods listed below, circle the ones that should have been declared `const`.

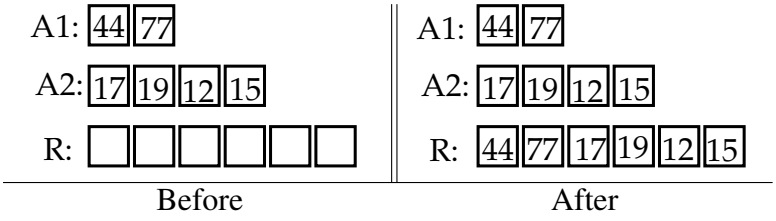
```
class bag {
public:
  bag();
  /* construct an empty bag*/

  void add(int item);
  /* PRE:  this bag is not full, and contains n elements: v1, v2, ...vn.
   * POST: this bag contains n+1 elements: v1,v2,...vn, item
   */

  bool contains(int item);
  /* PRE: none
   * POST: return true iff item is in this bag
   */

  size_t size();
  /* PRE : none
   * POST : returns the number of items in this bag.
   */

  void output();
  /* Writes the contents of the bag <v1,v2,...,vn> to the standard output
   * on one line separated by spaces.
   */

  int choose();
  /* PRE: this bag's size is not zero
   * An arbitrary element from the bag is removed and returned
   */

}
```

3. Complete the declaration of the class in the previous problem and provide implementations of the constructor; and the methods `add`, `contains`, and `size`.

4. Using one or more of the methods in the previous questions, write a standalone function that returns the smallest integer in a non-empty bag. You can use a temporary `bag`, just be sure that the content of the given bag is the same after the function runs.

```
int smallest(bag & b)
{









}
```

5. Suppose you have two arrays A1 and A2 of with N1 and N2 elements in each. Write a code fragment to concatenate A1 and A2, writing the result in array R. (You may assume that R is big enough.) An example is shown below.



6. Implement this method for array-based stacks. Assume `_data` is the instance variable containing the elements of the stack, `MAXSIZE` is the size of the array, and `_size` is the size of the stack. Recall this implies that the top stack element is at position `_size-1`.

```
void stack::pop(size_t k)
{
  // PRE:  Without loss of generality, this stack is non-empty,
  //       with elements ⟨V_1, V_2, ..., V_n⟩, with V_1 on top.
  //       1 ≤ k ≤ n.
  // POST: V_k is removed.









}
```