COMP 356 Homework Assignment 3

Acknowledgment. This assignment was written by Prof. Tim Wahls, with minor changes by John MacCormick.

1. (5 pts) Suppose that for a particular C++ compiler, the size of an `int` in memory is 2 bytes, of a pointer to `char` is 2 bytes and of a `double` is 8 bytes. How much memory is required for a variable of the following type `employeeType`? Show your work.

   ```
   typedef enum {ceo, manager, staff} employeeTag;

   typedef struct {
     char *name;
     double salary;
     char *address;
     employeeTag tag;
     union {
       struct {
         double goldenParachuteValue;
         int numberOfHouses;
       } ceoFields;
       struct {
         char *department;
         int teamSize;
       } managerFields;
       struct {
         char *department;
         int hireYear;
         char *speciality;
       } staffFields;
     } emp;
   } employeeType;
   ```

2. Consider the following C++ program:

   ```
   int main() {
     int *p1, *p2, *p3;
     int i = 3;

     p3 = &i;
     p1 = p3;
     p2 = new int;
     p1 = p2;
     delete(p1);
   }
   ```

   (a) (5 pts) Does this program create any garbage? Why or why not?

   (b) (3 pts) List all pointers that are dangling after the execution of `delete(p1);`.

3. (5 points) What are the relative advantages and disadvantages of garbage collection as compared to explicit deallocation of memory (by the programmer)?

4. (25 points) Implement a queue class in C++ as a template class, with the type of element stored in the queue as the parameter type. You must implement queues as linked lists (you are NOT allowed to use arrays).

   Your implementation must include the following member functions and constructors:

- a 0-arg constructor that initializes an empty queue
- a destructor which deallocates all memory associated with the queue
- `void enqueue(ElemType e)`, (where `ElemType` is the formal type parameter), which enqueues the element `e`. Your enqueue method must run in constant time (it can not scan the entire list).
- `ElemType dequeue()`, which dequeues and returns the next element from the queue. If the queue is empty, your member function should print an error message and not modify the queue. You can just throw an integer (i.e. just `throw 0;`) in this case. If the queue is not empty, the linked list node containing the dequeued element must be deallocated. Your `dequeue` member function must NOT create garbage. Your dequeue method must run in constant time (it can not scan the entire list).
- `bool isEmpty()`, which returns true if the queue is empty, and false otherwise
- `void print()`, which prints all elements of the queue

You must also include a definition of `main()` that creates at least two instances of your queue class (instantiated with different types) and uses those instances to invoke each of the above member functions (except for the destructor).

Hints:

- define an additional class that represents linked list nodes. If you nest the definition of this class inside the definition of your queue class, it can use the parameter type `ElemType` directly.
- your node class can NOT have a data member of the node class type directly (this is illegal in C++), but it can have a data member that is a pointer to the node class type
- do NOT define a destructor for your node class
- be careful not to dereference any dangling pointers in the `dequeue()` member function and in the destructor for the queue class
- you may find it easier to place all of your code in one .cpp file and to define all member functions directly inside their class (rather than splitting into a .h and a .cpp file). Many C++ compilers do not cleanly handle splitting a template class definition across a .h and a .cpp file.

Your source (.h, .cpp) file(s) must submitted to Moodle by the beginning of class on the due date. Submit all files as a single zip file.

Your program must compile and run correctly using `g++` on the Tome macs.

Your program will be graded on correctness, compliance with the above guidelines, use of encapsulation and coding style (including use of comments).