CPSC 221: Algorithms and Data Structures
Assignment #1, due Wednesday, 2014 May 28 at 17:00 (5pm) PST

Submission Instructions

Type or write your assignment on clean sheets of paper with question numbers prominently labeled. Answers that are difficult to read or locate may lose marks. We recommend working problems on a draft copy then writing a separate final copy to submit.

Each submission should include the names and student IDs of the authors at the top of each page. (You are strongly encouraged to work in pairs, but may not work in groups of three or more. Each pair submits a single copy of the assignment.) On your first page, also sign the statement "I have read and complied with the CPSC 221 2014S1 academic conduct policy as posted on the CPSC 221 course website." (See: `http://www.ugrad.cs.ubc.ca/~cs221/2014S1/syllabus.shtml#conduct`.) In keeping with the policy, you should also acknowledge on your first page any collaborators or resources that helped you with the assignment. Finally, staple your submissions pages together! We are not responsible for lost pages from unstapled submissions.

Submit your assignment to Box 31, in room ICCS X235. Late submissions are not accepted.

Questions

[10] 1. Stacks.

You are given a stack of integers. The stack has been implemented using an array. Assuming that the stack's initial state is given below,

(a) Draw the stack after each push operation has completed.

(b) If every time `top` is called, we also write down the integer value obtained, what list of numbers do we get?

Operations: `pop()`, `top()`, `pop()`, `push(5)`, `push(6)`, `top()`, `push(24)`, `top()`, `pop()`, `pop()`, `pop()`, `top()`, `push(7)`, `top()`.

[15] 2. Queues.

Consider the following C++ linked list implementation of a queue of integers (`int`).

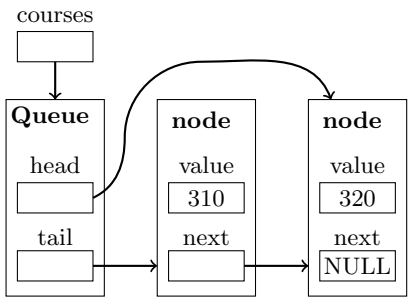
```
class Queue
{
   struct node {
      int value;
      node *next;
   };
   node *head = NULL, *tail = NULL;

   void enqueue(int n)
   {
      node *a = new node();
      a->value = n;
      if (head==NULL)
         tail = a;
      else
         head->next = a;
      head = a;
   }

   bool isEmpty()
   {
      return head==NULL;
   }

   int dequeue()
   {
      if (isEmpty())
         throw "Nothing to dequeue.";
      node *temp = tail->next;
      int n = tail->value;
      if (head==tail)
         head = NULL;
      delete tail;
      tail = temp;
      return n;
   }
};
```

The diagram below shows the state of memory just before some code executes:



Two **SUCCESSIVE** calls are made on the object pointed to by the pointer `courses`: `courses->dequeue()` and `courses->enqueue(221)`. For each operation, draw models representing the state of memory at the specified times.

(a) For the call to `courses->dequeue()`, draw the memory after the lines

i. `node *temp = tail->next;`
ii. `delete tail;`
iii. `tail = temp;`

(b) For the call to `courses->enqueue(221)`, draw the memory after the lines

i. `node *a = new node();`
ii. `a->value = n;`
iii. `head = a;`

[20] 3. Simplifying to Asymptotic Bounds.

For each of the following definitions of $T(n)$, give the big-Θ bound, as simplified as you can. You do not need to prove your result. However, to get partial credit for any minor mistakes, you should show how you arrived at your answer.

(a) **Example:** $T(n) = 42$, answer $\Theta(1)$.
(b) $T(n) = n^3 + n^2 + n$
(c) $T(n) = (n+4)(n+2) - n^2$
(d) $T(n) = \frac{n \log n}{2n} + n$
(e) $T(n) = \sum_{i=0}^{n-2}(3i+2)$
(f) $T(n) = \sum_{i=0}^{n-2}(3i+2)^2$
(g) $T(0) = 1$ and $T(n) = 2T(n-1) + 2$.

[15] 4. Comparing Asymptotic Behaviours.

(a) Let $f(n) = 8^{2\lg n}$ and $g(n) = 3n^7 + 7n$.

i. Asymptotically, does $f(n)$ or $g(n)$ grow faster? For this question, you may find it useful to use logarithm identities. Show your work for partial marks.
ii. Fill in the following blanks:
A. $f(n)$ is big-_____ of $g(n)$.
B. $g(n)$ is big-_____ of $f(n)$.

(b) Suppose a function $h(n)$ describing the runtime of one program is big-O of another function $k(n)$ describing the runtime of another program, can we say that at some point (for some sufficiently large input size n) the second program will take longer than the first? You may use examples in your answer.

[25] 5. Analyzing Runtime and Some Proofs.

Consider the following two functions `foo` and `bar` implemented in C++ below.

```
void foo(int n)
{
   for (int i=0; i<n, ++i)
   {
      if (i%2==0)
         cout << "foo" << endl;

      for (int j=0; j<n; ++j)
         cout << j << endl;
   }
}
```

```
void bar(int n)
{
   for (int i=0; i*i<n, ++i)
   {
      foo(n);

      for (int j=i; j<n; ++j)
         cout << j << endl;
   }
}
```

(a) Find an equation for the time complexity of each of the functions `foo` and `bar`. In your answer, you may assume that `cout` operations all take the same amount of time, and label this time with the constant c.

(b) Prove formally using the definition of big-O that `foo` has a runtime that is $O(n^3)$.

(c) Prove formally using the definition of big-Ω that `foo` has a runtime that is not $\Omega(n^{2.5})$.

(d) Prove formally using the definition of big-Θ that `bar` has a runtime that is $\Theta(n^{2.5})$.

[15] 6. Proof Practice.

Suppose you have two positive functions $f(n)$ and $g(n)$ where $f(n) \in O(h(n))$ and $g(n) \in O(k(n))$. If $k(n) \in O(h(n))$, show that $f(n) + g(n) \in O(h(n))$. (Hint: try combining the limits you get from $f(n) \in O(h(n))$, $g(n) \in O(k(n))$, and $k(n) \in O(h(n))$).