# Quiz 1 Solutions

Data Structures, 1999.

June 30, 1999

Total marks: 30

**Name: Theo Norvell**

**Student #:**

*Answer each question in the space provided or on the back of a page with an indication of where to find the answer.*

Linked lists are represented by pointers to nodes, with the NULL pointer (0) marking the end of the list. *Any nodes that are removed from the list should be deleted from the heap.* Nodes are defined as:

```
struct Node { int data ; Node* next; }
```

**Q0 [7]**

Consider:

```
static int count = 0 ;
int abc( int x ) {
      count += 1 ;
      if( x == 0 ) return 0 ;
      else if( x == 1 ) return 1 ;
      else {
           int s = abc( x / 2 ) ;
           int t = abc( (x+1)/2 ) ;
           return s+t+x ;} }
void main() { cout << abc( 32 ) << "\n" ; cout << count << "\n" ; }
```

What number will be printed by this program? *Hint: remember that integer division of positive numbers rounds down.*

*Solution 192 63*

*Explanation:*

*The subroutine computes a function. We can see that for an even $x$, $abc(x) =$*

$x + 2 \times abc(x/2)$. So

$$\begin{aligned}
& abc(32) \\
= \;& 32 + 2 \times abc(16) \\
= \;& 32 + 2 \times (16 + 2 \times abc(8)) \\
= \;& 32 + 2 \times (16 + 2 \times (8 + 2 \times abc(4))) \\
= \;& 32 + 2 \times (16 + 2 \times (8 + 2 \times (4 + 2 \times abc(2)))) \\
= \;& 32 + 2 \times (16 + 2 \times (8 + 2 \times (4 + 2 \times (2 + 2 \times abc(1))))) \\
= \;& 32 + 2 \times (16 + 2 \times (8 + 2 \times (4 + 2 \times (2 + 2 \times 1)))) \\
= \;& 192
\end{aligned}$$

*The global count variable counts the number of calls to abc. There is one call to abc(32), two to abc(16), four to abc(8), eight to abc(4), 16 to abc(2), and 32 to abc(1), for a grand total of 63.*

**Q1 [8]**

Consider

```
void xyz( char* p ) {
    if( *p != '\0' ) {
        cout << *p ;
        xyz( p+1 ) ;
        cout << *p ; } }
```

Give a reasonably complete pre- and postcondition for this subroutine. *Hint: Use words, not math to state the pre- and postconditions.*

*Solution:*

*Precondition: p is a pointer to the first character of a sequence of 1 or more characters ending with a nul byte. In other words, p points to a C-style string.*

*Postcondition: The sequence of characters (except for the terminating nul byte) will have been printed twice, first from front to back, and then from back to front.*

**Q2 [15]**

Given a possibly empty, null-terminated linked list, headed by in_head, and an integer, pivot, make a new list, headed by out_head, that contains copies of any nodes whose value is bigger than the given integer. The copied nodes should be in the same order as the original nodes. E.g. if the original list contains $\langle 3, 1, 4, 2, 3, 5 \rangle$ and the integer is 2 then the new list will contain $\langle 3, 4, 3, 5 \rangle$.

Set ok to true if you succeed and to false if you run out of memory.

*Use recursion. Note that out_head is an output parameter.*

*Solution:*

```
#include <new>
using namespace std;
...
```

```
void copy_bigger( Node* in_head, int pivot, Node* &out_head, bool &ok ) {
      if( in_head == 0 ) {
           ok = true ; }
      else if( in_head->data > pivot ) {
           out_head = new(nothrow) Node ;
           if( out_head ) {
                out_head->data = in_head->data ;
                copy_bigger( in_head->next, pivot, out_head->next, ok ) ;
           else {
                ok = false ; } }
      else {
           copy_bigger( in_head->next, pivot, out_head, ok ) ; }
}
```

---