CS 216 Exam 1 – Fall 2005 – Solution

Name:_______________________ Lab Section: _________________

Email Address:_______________

This exam is closed note, closed book. You will have an hour and fifty minutes total to complete the exam. You may NOT use calculators.

It is an Honor Code violation to discuss this exam with ANYONE (including other students who have already taken the exam) until after 9:30pm on the exam day.

Good Luck!!

<table>
<thead>
<tr>
<th></th>
<th>MAX</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Write and sign pledge after taking the exam:
1) (3 points) Write the C++ expression that defines an STL vector that will hold string values. \textit{Also}, add the string “hello” to the end of this vector. \textit{Also}, write the C++ expression that gives the number of elements stored in your vector.

\begin{verbatim}
vector<string> v;
v.push_back(“hello”);
v.size()  // or any expression that includes this
\end{verbatim}

2) (5 points) What does “ADT” stand for? Explain the concept of an ADT, using a good example to illustrate your definition.

\textit{ADT stands for “abstract data type”. An ADT describes a model for storing data (often a collection of items) and a set of operations that can be performed on this data. Examples:}
- The List ADT is a sequence of items, with operations like: get the first item; get the next item; retrieve by index; size; etc.
- The Stack ADT is a set of items accessed in a last-in, first-out manner, with operations like: push; pop; etc.
- The Queue ADT is a set of items accessed in a first-in, first-out

3) (3 points) Name the 3 functions that make up the “Gang of Three” (also known as “the Big Three”) in C++.

\textit{destructor; copy constructor; assignment operator (AKA operator=)}

4) (6 points) Choose two of Gang of Three functions and explain what situation makes it important for you to implement these 3 functions yourself and what can go wrong if you do not. \textit{Also}, for the two functions you choose, give an overview of what your implementation of the function must do. (Your overview does not have to have every detail, but it must be clear you understand why they must be written and how.)

**Destructor:** if the class’ data-member(s) references dynamic memory (created with new), then you must implement a destructor that properly uses delete to free-up all dynamic memory when an object is de-allocated. Otherwise you’ll have a memory leak.

**Copy constructor:** Again, you must write this when data-members reference dynamic memory. Otherwise you get a shallow-copy when C++ needs to use the copy constructor, e.g. call-by-value, function returns an object, defining an object like: Foo x(y);

**Assignment operator:** Again, you must write this when data-members reference dynamic memory to avoid getting a shallow-copy when you assign an object to another. It must check for self-assignment, destroy the current data in the object, make sure there’s room, and copy the data into the new object.
5) Here are two questions on postfix expressions:

(a) (2 points) What is the value of this postfix expression: \( \text{5 1 \text{--} 3 * 3 \text{--} 1 * /} \)

The value is: 3  
The infix of this is: \( ((5 - 1) * 3) / (3 - 1) * 2) \)

(b) (2 points) When evaluating this expression using a stack, show the contents of the stack after the three operators have been “executed” (in the proper order that they would be executed when processing the postfix expression). Make it clear what the top of the stack is.

The stack contains two values: 12 and 2 (the top). This is after doing one subtraction (5-1), one multiplication (4*3), and another subtraction (3-1).

6) (3 points) Name 3 factors that are ignored by big-Oh notation

Hardware, Operating system, compiler, programming language used, assumes that all operations take the same amount of time. Other answers that were accepted: ignores constant factors, throws out lower ordered terms, (these were described as big-Oh math), also does not work for small values of N (although this is not really a factor that it ignores, big-Oh just doesn’t give you a good estimate of run-time/memory use for small values of N)

7) (2 points) Order the following rates of growth from slowest rate of growth to fastest rate of growth:

\( n^{20} \), 25, \( \log n \), \( n \log n \), \( n \), \( 2^n \), \( 10^5 \)

\( 25, 10^5=100000 \) (both are \( \Theta(1) \)); \( \log n \); \( n \); \( n \log n \); \( n^{20} \); \( 2^n \)

8) Here are questions on order-class definitions:

a) (2 points) Complete this definition: \( O(f(n)) \) is the set of functions that…

We accepted either of these answers:

… that grow at the same rate or more slowly than \( f(n) \). Or,

… whose value is \( \leq c f(n) \) for all \( n > N_0 \) for some positive values of \( c \) and \( N_0 \)

b) (2 points) Explain the difference between \( O(f(n)) \) and \( \Theta(f(n)) \).

\( \Theta(f(n)) \) is the set of functions that grow at the same rate as \( f(n) \).
9) (10 points total) Describe the running time of the following pseudocode using Big-Theta or Big-Oh notation in terms of the variable \( n \) (where \( n \geq 0 \)). In using Big-Theta or Big-Oh, make the most precise statement about the running time that’s possible. Assume all variables used have been declared. \textit{Show your work for partial credit.}

\begin{verbatim}
int foo(int k) {
    int sum;
    for (int i = 0; i < k%100; ++i) {
        sum = sum + (i * i);
    }
    return sum;
}

int bar(int k) {
    int sum = 0;
    if (k < 100) {
        for (int i = 0; i < k; ++i) sum += i;
    }
    return sum;
}

a) answ = foo(n);   \( \Theta(1) \) because above 0 \( k \% 100 \leq 99 \)

b) answ = bar(n);   \( O(n) \) is the best answer. If \( n < 100 \), it’s \( \Theta(1) \). If \( n \geq 100 \), it’s \( \Theta(n) \). In all cases it grows slower than \( n \) so \( O(n) \) describes all cases.

c) for (int j = 0; j < n; ++j) { \( O(n^2) \)
    cin >> val;
    for (int i = 0; i < j; ++i) {
        b = b * val;
        for (int k = 0; k < 10; ++k)
            c = b + c;
    }
}

d) for (int j = 0; j < n; ++j) { \( O(n^3) \)
    for (int i = 0; i < n; ++i) {
        b = b * j;
        for (int k = 0; k < i; ++k)
            c = b + c;
    }
}

e) for (int i = 0; i < n; ++i) { \( O(n^2) \)
    for (int j = 1; j <= i ; ++j){
        sum = sum + i;
    }
    for (int k = 0; k < n + 1000; ++k){
        a[k] = a[k] + sum;
    }
}
\end{verbatim}
10) (4 points total) What is the representation of each of the following in the indicated radix? Be sure to show your work.

a) 1325 in decimal

\[1 \times 5^2 + 3 \times 5^1 + 2 \times 5^0 = 25 + 15 + 2 = 42\]

b) 1AF16 in octal

The easiest way to do this is to write it in binary and then re-write it in octal:

\[1AF_{16} = 0001 1010 1111 \text{ binary or } 000 110 101 111 \text{ (same bits, grouped by 3 digits)} = 0657_8\]

11) (6 points total) Consider the positive binary integer represented in two’s complement:

\[0101001001111101_2\]

a) Express this binary number in octal

\[0101001001111101_2 = 0 101 001 001 111 101 = 051175_8\text{ or } 51175_8\]

b. Express this binary number in hexadecimal

\[0101001001111101_2 = 0101 0010 0111 1101 = 527D_{16}\]

c. Negate the number (i.e. give the two’s complement representation of a negative version of the same number) Use the same number of bits.

\[
\begin{align*}
\text{Complement first:} & \quad 0101001001111101 \\
\text{Then add one:} & \quad + 1010110110000010 \\
\text{ANSWER:} & \quad 1010110110000011
\end{align*}
\]
12) (3 points) If you have N bits to store a two’s complement number, write expressions for the largest positive value and the smallest negative value that can be represented.

Largest positive: $2^{n-1} - 1$  
Smallest negative: $-2^{n-1}$

13) (7 points) Assume we are using the 32-bit IEEE single precision floating point format as described in class and used in lab. The mantissa has 24 bits including the hidden bit. There is one sign bit and there are eight exponent bits. The exponent is stored in excess 127.

What decimal floating point number is represented by the following 32 bits? SHOW YOUR WORK!

1100 0001 1000 1110 0000 0000 0000 0000

a) Is this a positive or negative number? Negative, since the sign-bit is one

b) What is the exponent (in base 10)?

$10000011 = 128 + 2 + 1 = 131$ in excess-127. So this is really $131 - 127 = 4$.

c) What is the value of the mantissa (in base 10)?

The mantissa/significand is 00011100000000000000000, and when we add the hidden bit it’s 1.0001112 or

$1 + (0 \times \frac{1}{2}) + (0 \times \frac{1}{4}) + (0 \times \frac{1}{8}) + (1 \times \frac{1}{16}) + (1 \times \frac{1}{32}) + (1 \times \frac{1}{64})$

Well, I will admit I shouldn’t have made you deal with this much division! This is equal to $1 + 7/64$ which is $1.109_{10}$ but we were liberal with grading this if you got this far.

d) What is the total value?

Note: you may leave your answer in the form: $value_{10} \times base^{exponent}$

Putting it all together: $-1.109_{10} \times 2^4$

This is -17.75, one of the examples we did for you in the email on this.
14) (20 points) This question tests your understanding of queues. You must implement a queue ADT in C++. The underlying representation of the queue should be a C++ primitive array. Your queue should store integers and should handle errors (printing an error message is fine). After any number of enqueue/dequeue operations in any order, it should always be able to hold the maximum number of items.

You will be graded mostly on the correctness of the ideas of your solution rather than exact C++ syntax, but your solution should be clear. Correct C++ code is the best way to ensure we understand your solution. You may NOT use the STL in any way for this question. You should use the header file provided below.

You should implement all the functions with \( \rightarrow \) in front of them.

class Queue {
public:
\( \rightarrow \) Queue(); // constructor
\( \rightarrow \) void enqueue(int val); // inserts val onto the queue.
\( \rightarrow \) int dequeue(); // returns and removes the value on
// the front of the queue.
\( \rightarrow \) bool isEmpty(); // returns true if the queue contains
// no elements.
\( \rightarrow \) bool isFull(); // returns true if the queue is full
const int MAX_SIZE = 100; // max number of elements that
// can be stored in the queue
private:
// Add more data members here if needed.

    int data[MAX_SIZE];

    int front; // index to the front of the queue
    int back;  // index to the back of the queue
    int size;  // how many items in the queue
};

Notes: See Weiss’ Data Structures book. The key point about implementing queues is to have efficient access to the front of the queue (for accessing or removing the front item) and the rear of the queue (for inserting a new item). For a linked-list implementation, we use two pointers. For an array-based implementation, we use two int variables to access the front and rear.
Queue::Queue() { // constructor: worth 3 points
    size = 0;
    front = 0;
    back = -1;
}

bool Queue::isEmpty() { // worth 2 points
    return size == 0;
}

bool Queue::isFull() { // worth 2 points
    return size == MAX_SIZE;
}

void Queue::enqueue(int val) { // worth 4 points
    if (isFull())
        cout << "Queue is full!" << endl; // or something
    else {
        ++back; // move to next available position
        // too far? set back to beginning of array
        if (back == MAX_SIZE) back = 0;
        data[back] = val;
        ++size;
    }
}

int Queue::dequeue() { // worth 5 points
    if (isEmpty()) {
        cout << "Queue is empty!" << endl; // or something
        return -999; // or something
    }
    else {
        int val = data[front];
        ++front;
        // too far? set front to beginning of array
        if (front == MAX_SIZE) front = 0;
        --size;
        return val;
    }
}
15) (2 points) List the names of any of the Stack Queue member functions you wrote that have time-complexity that is worse than \( \Theta(n) \), where \( n \) is the number of items in the queue.

None should be if you implemented Queue properly. (That’s the reason it’s implemented that way.)

(If you didn’t implement Queue right and your answer here is correct for your solution to #14, you shouldn’t lose points here.)

The remaining questions refer to these declarations:

```cpp
class Node {
    public:
        string Value;
        Node *Next;
};
Node *Temp1;
Node Temp2;
Temp1 = new Node;
```

16) [2 points each, 8 points total] As you can see, both Temp1 and Temp2 can be used to access a Node object and the data-members inside the object. Answer the following questions by writing a simple C++ statement or expression. (If it helps you to answer the questions, feel free to draw a picture.)

a) Use Temp1 to assign the string "hello" to the Node object's Value data-member.

```
Temp1->Value = "hello";
```

b) Use Temp2 to assign the string "hello" to the Node object's Value data-member.

```
Temp2.Value = "hello";
```

c) Write a line of code that stores the memory location of the string inside the Temp2 Node object into a variable named foo. You must give a C++ definition for this variable foo that will hold the memory location of this string.

```
string *foo = &Temp2.Value;
```

d) Assume that the object that Temp1 refers to has valid data-values in its two data-members. Write the C++ expression that gives us the Node object that Temp1’s Next data member references.

```
*(Temp1->Next)
```
17) [4 points] Assuming that Temp1 points to a list of Nodes that looks like this:

```
      Temp1
  +---+-----+-----+-----+
  |   |     |     |     |
  +---+-----+-----+-----+
       5         6         12
```

Write code that will insert a new Node containing the value 7 into the list after the Node containing 6 and before the Node containing 12. This new node should be allocated on the heap. After inserting the value 7 into the list, your list should appear as in the following problem below. You do not need to write code that searches for the values 6 or 12 – just use pointer values shown in the picture. Please ask the instructor if you are unsure what is being asked for in this question. Feel free to declare more variables as needed.

```
Node *Temp1;
// code deleted that builds the list as shown above

// your code here:

Node *p;
p = Temp1->Next;  // p points to 2\text{nd} node

Node *TempNode = new Node();
TempNode->Value = "7";

TempNode->Next = p->Next;  // make new node’s Next point
                          // to node after 2\text{nd} node

p->Next = TempNode;        // 2\text{nd} node’s Next points to
                          // new Node.
```
18) [4 points] Assuming that Temp1 points to a list of Nodes that looks like this:

```
18) [ 4 points] Assuming that Temp1 points to a list of Nodes that looks like this:

   Temp1

   [5] → [6] → [7] → [12] →

   Write code that will remove the first Node in the list and return its memory to the heap. After removing the node, your list should look like the picture shown below. Feel free to declare more variables as needed.

   Node *Temp1;
   // code deleted that builds the list as shown above
   ... . . . .
   // your code here:

   Node *p;
p = Temp1;  // p points also points to the first node

   Temp1 = Temp1->Next;  // Temp1 to points to node after its current value

   delete p;
```

**After:**

```
After:

   Temp1

   [6] → [7] → [12] →
```