

CS 14 – Basic Proficiency Exam

20 points possible

1. (3 pts) Explain when to use the dot (.) operator and when to use the arrow (→) operator.

2. (2 pts) Is it a good idea to make a class destructor private? Why or why not?

3. (1 pt) How many times does the “fib” function get called in the following code?

```
int main ( ) {
    cout << fib ( 5 );
    return 1;
}

int fib ( int x ) {
    return fib(x-1) + fib(x+2);
    if ( x <= 1 ) return 1;
}
```

4. (1 pt) How many times does the “factorial” function get called in the following code?

```
int main ( ) {
    factorial ( 5 );
    return 1;
}

int factorial ( int x ) {
    if ( x <= 1 ) return 1;
    else return x * factorial(x-1);
}
```

5. (2 pts) What gets printed in the following code?

```
int main ( ) {
    cout << maconacci ( 5 );
    return 1;
}

int maconacci ( int x ) {
    if ( x == 0 ) return 1;
    if ( x == 1 || x == 2 ) return 3;
    return maconacci (x-1) * maconacci (x-2) + maconacci (x-3);
}
```

6. (2 pts) What gets printed in the following code?

```
int main ( ) {
    int x = 5, y = 10;
    swap ( x, y );
    cout << x << “ “ << y << endl;
    return 1;
}

void swap ( int x, int y ) {
    int temp = x;
    x = y;
    y = temp;
}
```

7. (4 pts) Given the following snippet of code:

```
int* x = new int;  
int y = *x;  
int* z = &y;  
y = 10;  
int a = *z;  
(*x) ++;  
(*z) --;
```

Describe what the following statements will produce (either what will be printed or what will happen). If a memory address is printed, be as specific as you can by telling me what variable/value the memory address is associated with.

a) `cout << x << endl;`

b) `cout << *x << endl;`

c) `cout << y << endl;`

d) `cout << &y << endl;`

e) `cout << z << endl;`

f) `cout << *z << endl;`

g) `cout << a << endl;`

8. (5 pts) Create a class called **IntegerSet**. **IntegerSet** will have two member variables. One data member variable, **set**, will be a pointer to a dynamically created integer array. The second data member variable, **size**, will hold the size of the **set** array. Create a constructor that takes as a parameter the size of the **set** array and the array is dynamically allocated in the constructor and all array locations are initialized to 0. Write a destructor to delete all allocated memory. You must write the code for the class and the code for the constructor and destructor. Remember to provide error checking and use good programming constructs throughout. Syntax will be graded.