Exam Reference

main function template from public course web page:

```
/*-----
  Signature: main: void -> int

  Purpose: either:
           <describe the program being written> OR
           testing program for the functions <f1>, <f2>, ...

  Examples: <describe, in prose, what the effect of running this
             main() function should be>

  by:
  last modified:
-----*/

#include <iostream>
#include <string>
// #include "something.h"
using namespace std;

int main()
{
    // do something

    return EXIT_SUCCESS;
}
```

WITHIN FILE boa.h:

```
#ifndef boa_h
#define boa_h

/*-------------------------------------
   a boa is a class instance:
      boa(string color, double length,
              string food)
   ...representing a boa with:
      a color that is its primary coloring,
      a length in meters,
      a preferred food

template for a function with a boa parameter a_boa:

ret_type process_boa(boa a_boa)
{
    return ... a_boa.get_color() ...
           ... a_boa.get_length() ...
           ... a_boa.get_food() ... ;
}
--------------------------------------*/

#include <string>
using namespace std;

class boa
{
    public:
        // constructors

        boa(string a_color, double a_length, string a_food);
        boa( );

        // selectors

        string get_color( ) const;
        double get_length( ) const;
        string get_food( ) const;

        // modifiers
        void set_length(double new_length);
        void set_color(string new_color);
        void set_food(string new_food);

        // other methods

        bool longer_than(boa another_boa) const;
        bool longer_than(double given_length) const;

    private:
        string color;
        double length;
        string food;
};

#endif
```

WITHIN FILE boa.cpp:

```
/*--------------------------------------------------
   a boa is a class instance:
      boa(string color, double length,
              string food)
   ...representing a boa with:
      a color that is its primary coloring,
      a length in meters,
      a preferred food

template for a function with a boa parameter a_boa:

ret_type process_boa(boa a_boa)
{
    return ... a_boa.get_color() ...
           ... a_boa.get_length() ...
           ... a_boa.get_food() ... ;
}
--------------------------------------------------*/

#include "boa.h"
using namespace std;

// constructors

boa::boa(string a_color, double a_length, string a_food)
{
    color = a_color;
    length = a_length;
    food = a_food;
}

boa::boa( )
{
    color = "green";
    length = 6;
    food = "alligators";
}

// selectors

string boa::get_color( ) const
{
    return color;
}

double boa::get_length( ) const
{
    return length;
}

string boa::get_food( ) const
```

```
{
    return food;
}

// modifiers

void boa::set_length(double new_length)
{
    length = new_length;
}

void boa::set_color(string new_color)
{
    color = new_color;
}

void boa::set_food(string new_food)
{
    food = new_food;
}

// other methods

// signature: boa::longer_than: boa -> bool
// purpose: expects another boa, and produces whether the
//          calling boa is longer in length than the given
//          ther boa
// examples:
//     boa george("purple", 75, "dragons");
//     boa carol("green", 6, "alligators");
//     carol.longer_than(george) == false
//     george.longer_than(carol) == true
//     george.longer_than(george) == false

bool boa::longer_than(boa another_boa) const
{
    return length > another_boa.length;
}

// signature: boa::longer_than: double -> bool
// purpose: expects a length in meters, and produces whether
//          the calling boa is longer than this given length
// examples:
//     boa carol("green", 6, "alligators");
//     carol.longer_than(7) == false
//     carol.longer_than(6) == false
//     carol.longer_than(5) == true

bool boa::longer_than(double given_length) const
{
    return length > given_length;
}
```

WITHIN FILE boa_test.cpp:

```
#include <iostream>
#include <cmath>
#include "boa.h"
using namespace std;

/*---------------------------------------------------
 Signature: boa_test : void -> bool
 Purpose: expects nothing, and produces whether boa's
         selectors return what is expected for example
         boas, whether its modifiers have the expected
         effect, and whether its other functions produce
         what they should
 Examples: boa_test( ) == true
--------------------------------------------------*/

bool boa_test( )
{
    boa george("fuschia", 50, "crocodiles");
    boa carol;

    // these are to keep track of my test results

    bool selector_results;
    bool modif_results;
    bool other_results;

    selector_results =
           (george.get_color() == "fuschia") and
           (george.get_length() == 50) and
           (george.get_food() == "crocodiles") and
           (carol.get_color() == "green") and
           (carol.get_length() == 6) and
           (carol.get_food() == "alligators");

    // exercise the modifier methods

    george.set_color("purple");
    george.set_length(75);
    george.set_food("dragons");

    modif_results =
           (george.get_color() == "purple") and
           (george.get_length() == 75) and
           (george.get_food() == "dragons");

    other_results = (carol.longer_than(george) == false) and
                    (george.longer_than(carol) == true) and
                    (george.longer_than(george) == false) and
                    (carol.longer_than(7) == false) and
                    (carol.longer_than(6) == false) and
                    (carol.longer_than(5) == true);

    return (selector_results and modif_results and other_results);
}
```