CISS245: Advanced Programming
Final f01

Name: bglandis1@cougars.ccis.edu Score:

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

When submitting using alex, enter `f01` for work.

Instructions

- This is a closed-book, no-discussion, no-calculator, no-browsing-on-the-web no-compiler/no-MIPS-simulator test.
- Cheating is a serious academic offense. If caught you will receive an immediate score of -100%.
- If a question asks for a program output and the program or code fragment contains an error, write `ERROR` as output. When writing output, whitespace is significant.
- If a question asks for the computation of a value and the program or code fragment contains an error, write `ERROR` as value.
- When you're asked to write a C++ statement, don't forget that it must end with a semicolon.
- Bubblesort refers to the bubblesort algorithm in our notes where values are sorted in ascending order.

Honor Statement

I, BRYSEN LANDIS, attest to the fact that the submitted work is my own and is not the result of plagiarism. Furthermore, I have not aided another student in the act of plagiarism.

Question	Points
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Question	Points
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	

TOTAL	

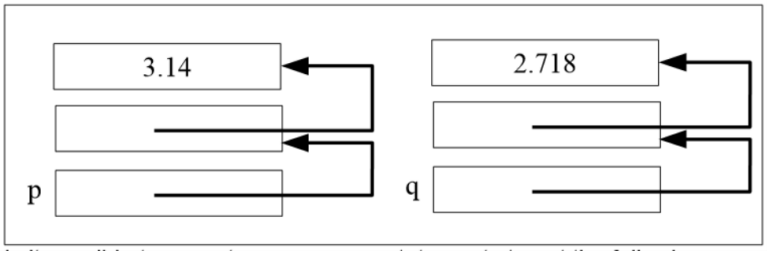
Q1. What is the output of this code fragment if the user enters `0` for `i`, `1` for `j`, and `2` for `k`?

```
int i;
int j;
int k;
std::cin >> i >> j >> k;
int * p = &i;
int * q = &j;
int * r = &k;
q = p;
p = r;
std::cout << *p << ' ' << *q << ' ' << *r << '\n';
```

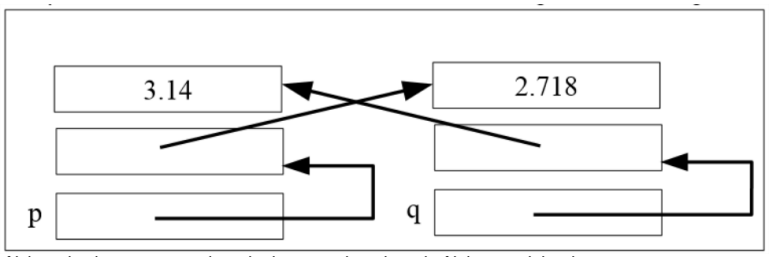
Answer:

```
2 0 2
```

Q2. At one point in the execution of a program, you have the following memory model:



Is it possible to execute one or more statements to get the following:



If it is, write the statements (use the least number please). If it isn't explain why. Answer:

```
int ** t = p;
p = q;
q = t;
```

Q3. The following code fragment has a memory leak. Correct the code to ensure there's no memory leak. (Make corrections to the code below directly.)

Answer:

```
int f(int n)
{
    int * a = new int[n];
    int * i = new int;

    a[0] = rand();

    for (*i = 1; *i < n; ++(*i))
    {
        a[*i] += rand();
        a[*i] -= a[*i - 1];
    }
    int ret = a[n - 1];

    delete[] a;
    delete i;

    return ret;
}
```

Q4. The following works with a 2D array that is in the heap. (Look at the double for-loop.) Complete the declaration of `p`, allocate memory appropriately, and at the end of the code fragment, deallocate memory used.

Answer:

```
// Declare p and allocate memory for p.

int ** p = new int*[10];
for (int i = 0; i < 10; ++i)
{
    p[i] = new int[20];
}

for (int r = 0; r < 10; ++r)
{
    for (int c = 0; c < 20; ++c)
    {
        p[r][c] = 42;
    }
}

// Deallocate memory use by p.

for (int i = 0; i < 10; ++i)
{
  delete[] p[i];
}

delete[] p;
```

Q5. The following program computes and prints the minimum of an array of 10 integer values entered by the user. Follow the instructions (in comments) in order to complete the program. Do not remove the comments. I have already declared all the variables you need. Therefore do not declare any other variables. You must allocate and deallocate memory correctly.

Answer:

```
#include <iostream>

int main()
{
    int * p;
    int * q;
    int * min;

    // Allocate memory for p so that p points to an array of 10 values
    // in the memory heap.

    p = new int[10];

    // Prompt user for 10 integers and put them in the array that p
    // points to.
    for (q = p; q < p + 10; ++q)
    {
      std::cin >> *q;
    }

    // Compute the minimum of values in the array of 10 values that p
    // points to and store in integer that min points to.
    min = p;
    for (q = p + 1; q < p + 10; ++q)
    {
        if (*q < *min)
        {
            min = q;
        }
    }

    std::cout << *min << std::endl;

    // Deallcate memory used by p
    delete[] p;
    return 0;
}
```

Q6. Write a function that checks if a string is a palindrome. A palindrome is a string that is it's own "reverse". In other words, you read the string left-to-right and right-to-left, you get the same string. For instance

`"madam"`

Another example is

`"tacocat"`

Write a function `is_two_palindromes` that checks if string is made up of two palindromes. For instance

`"madammom"`

is made up of the concatenation of two palindromes: `madam` and `mom`.

Don't forget that a C-string is null-terminated, i.e., there's an end-of-data character of `'\0'` in the string that indicates where the string data ends. You *cannot* use any C-string functions or the C++ string class. You can add as many functions as you like. I've added an `is_palindrome` function you can choose to use that or not.

Answer:

```
bool is_palindrome(char * start, char * end)
{
  end--;
  while (start < end)
  {
    if(*start != *end) return false;
    start++;
    end--;
  }

  return true;

}

bool is_two_palindromes(char s[])
{
  char * end = s;
  while (*end != '\0') end++;

  for (char * mid = s + 1; mid < end; mid++)
  {
    if (is_palindrome(s, mid) && is_palindrome(mid, end))
    {
      return true;
    }
  }
```

```
  return false;
}
```

Q7. Complete the following `Rectangle` class given the following requirements:

- There are two private member variables `width_` and `height_`, both of `double` type.
- You can construct a `Rectangle` object of a given width (say 10) and given height (say 5) as follows:

```
Rectangle r(10, 5);
```

  You must use an initializer list whenever possible.
- You can also construct a `Rectangle` object (say `r1`) with another `Rectangle` object (say `r2`) so that `r1` and `r2` have the same width and height:

```
Rectangle r1(r2);
```

  You must use an initializer list whenever possible.
- You can get the value of the `width_` of a Rectangle object (say `r`) as follows:

```
std::cout << r.get_width() << std::endl;
```

- You can also get the value of the `width_` of a `Rectangle` object (say `r`) as follows:

```
std::cout << r.width() << std::endl;
```

- You can set the value of the `width_` of a `Rectangle` object (say `r`) as follows:

```
r.set_width(12.34);
```

- You can also set the value of the `width_` of a `Rectangle` object (say `r`) as follows:

```
r.width() = 12.34;
```

- You can also get the area of a `Rectangle` object (say `r`) as follows:

```
std::cout << r.area() << std::endl;
```

All methods are to be implemented in the header file below.

(Turn page)

Answer:

```
// file: Rectangle.h

#ifndef RECTANGLE_H
#define RECTANGLE_H

class Rectangle
{
  public:
  Rectangle(double w, double h)
  : width_(w), height_(h)
  {}

  Rectangle(const Rectangle & r)
  : width_(r.width_), height_(r.height_)
  {}

  double get_width() const
  {
    return width_;
  }

  double width() const
  {
    return width_;
  }

  double & width()
  {
    return width_;
  }

  void set_width(double w)
  {
    width_ = w;
  }

  double area() const
  {
    return width_ * height_;
  }

  private:
  double width_;
  double height_;
};

#endif
```

Q8. It has been decreed that every living thing in the Milky Way galaxy must have an integer `id` (an integer). A human being of course is a living thing, but a human being has a gender (a character) and number of heads, arms, and legs. Create three classes: `LivingThing`, `Earthling` and `Martian` where `Earthling` is a subclass of `LivingThing`. You only need to have enough code for the following code to execute:

```
// john has id 3423452, is male, has 2 heads, 3 arms, and 4 legs
Earthling john(3423452, 'm', 2, 3, 4);
```

A martian also has an id, but no gender, have heads and legs, but no arms. All member variables must be private. Furthermore, based on the information above, various private member variables must be placed in the right class to minimize code duplication

Implement the `LivingThing` class below. (See next questions for the `Earthling` and `Martian` class.)

Answer:

```
// file: LivingThing.h

#ifndef LIVINGTHING_H
#define LIVINGTHING_H


class LivingThing
{
  public:
  LivingThing(int id) : id_(id) {}

  private:
  int id_;
};

#endif
```

Q9. This is a continuation of the previous question. Implement the `Earthling` class below.

Answer:

```
// file: Earthling.h
#ifndef EARTHLING_H
#define EARTHLING_H

#include "LivingThing.h"

class Earthling : public LivingThing
{
  public:
  Earthling(int id, char gender, int heads, int arms, int legs)
  : LivingThing(id), gender_(gender), heads_(heads), arms_(arms), legs_(legs)
  {}

  private:
  char gender_;
  int heads_;
  int arms_;
  int legs_;
};

#endif
```

Q10. This is a continuation of the previous question. Implement the `Martian` class below.

Answer:

```
// file: Martian.h
#ifndef MARTIAN_H
#define MARTIAN_H

#include "LivingThing.h"

class Martian : public LivingThing
{
  public:
  Martian(int id, int heads, int legs)
  : LivingThing(id), heads_(heads), legs_(legs)
  {}

  private:
  int heads_;
  int legs_;
};

#endif
```

Q11. The following class allows us to create objects that models rational numbers:

```
class Rational
{
public:
    Rational(int n, int d)
        : n_(n), d_(d)
    {}
    // other methods not shown
private:
    int n_;
    int d_;
};
```

Complete two methods that allows you to perform additions (`+` and `+=`). As an example, the following usage of the Rational class should work correctly:

```
Rational r0(1, 2), r1(3, 4); // r0 models 1/2 and r1 models 3/4
Rational r2 = r0 + r1;       // r2 models 1/2 + 3/4 ,
                             // i.e., (1 * 3 + 2 * 4) / (2 * 4)
Rational r3 = r2;
r3 += r2;                    // r3 models r3 + r2
```

Your code *must* be minimal. (HINT: `+` should use `+=`.)

(Turn page)

Answer:

```
// file: Rational.h

#ifndef RATIONAL_H
#define RATIONAL_H

class Rational
{
public:
    Rational(int n, int d)
        : n_(n), d_(d)
    {}

    Rational & operator+=(const Rational & r)
    {
      n_ = n_ * r.d_ + d_ * r.n_;
      d_ = d_ * r.d_;
      return *this;
    }

    Rational operator+(const Rational & r) const
    {
      Rational result = *this;
      result += r;
      return result;
    }

private:
    int n_;
    int d_;
};

#endif
```

Q12. The following class allows us to create objects that handle dynamic array of integers, i.e., each object contains a pointer to an array in the heap:

```
class IntDynArr
{
public:
    IntDynArr(int size)
        : p_(new int[size]), size_(size), capacity_(size)
    {}
    // other methods not shown
private:
    int * p_;
    int size_;
    int capacity_;
};
```

Of course since objects of the class uses a resource that is not automatically released, you have to write the

- Destructor
- Copy constructor
- Assignment operator

in order to overwrite the default ones:

```
class IntDynArr
{
public:
    IntDynArr(int size)
        : p_(new int[size]), size_(size), capacity_(size)
    {}

    ~IntDynArr();
    IntDynArr(const IntDynArr &);
    IntDynArr & operator=(const IntDynArr &);

    // other methods not shown
private:
    int * p_;
    int size_;
    int capacity_;
};
```

Implement the destructor below. (The copy constructor and assignment operator are in the next two questions.)

Answer:

```
// file: IntDynArray.cpp

#include "IntDynArray.h"

IntDynArray::~IntDynArr()
{
  delete[] p_;
}

// other methods not shown
```

Q13. This is a continuation of the previous question. Implement the copy constructor below for `IntDynArray`. (Constructors not using initializer lists as much as possible are considered incorrect.)

Answer:

```
// file: IntDynArray.cpp

#include "IntDynArray.h"

IntDynArray::IntDynArr(const IntDynArr & x)
: p_(new int[x.capacity_]), size_(x.size_), capacity_(x.capacity_)
{
  for (int i = 0; i < size_; ++i)
  {
    p_[i] = x.p_[i];
  }
}

// other methods not shown
```

Q14. This is a continuation of the previous question. Implement the assignment operator below for `IntDynArray`.

Answer:

```
// file: IntDynArray.cpp

#include "IntDynArray.h"

IntDynArray & IntDynArray::operator=(const IntDynArr & x)
{
  if (*this != &x)
  {
    delete[] p_;
    p_ = new int[x.capacity_];
    size_ = x.size_;
    capacity_ = x.capacity_;
    for (int i = 0; i < size_; ++i)
    {
      p_[i] = x.p_[i];
    }
  }
  return *this;
}

// other methods not shown
```

Q15. Complete the following recursive function `sumsquares()` so that `sumsquares(n)` computes $0^2+1^2+2^2+\cdots+n^2$, i.e., it computes the "sum of squares". n is an integer that is at least 0.

Answer:

```
int sumsquares(int n)
{
  if (n = 0) return 0;
  return n * n + sumsquares(n - 1);
}
```

Q16. Write a recursive function to print the integers from address `start` up to *but not including* address `end`. Between every two integers, the function prints a space. After all integers are print, the function prints a newline character.

Answer:

```
void println(int * start, int * end)
{
  if (start >= end)
  {
    std::cout << '\n';
    return;
  }
  std::cout << *start;
  if (start + 1 < end)
  {
    std::cout << ' ';
  }
  println(start + 1, end);
}
```

Q17. Complete the following function that performs bubblesort on `*start`, `*(start+1)`, ..., `*(end-1)` where `x` is an array of integer values. Both functions must be recursive.

Answer:

```
// Performs one pass of bubblesort on *start, *(start + 1), ...,
// *(end - 1).
void bubblesort_pass(int * start, int * end)
{
  if (start + 1 >= end) return;
  if (*start > *(start + 1))
  {
    int t = *start;
    *start = *(start + 1);
    *(start + 1) = t;
  }
  bubblesort_pass(start + 1, end);
}

// Performs bubblesort on *start, *(start + 1), ..., *(end - 1).
// Uses bubblesort_pass.
void bubblesort(int * start, int * end)
{
  if (start >= end) return;
  bubblesort_pass(start, end);
  bubblesort(start, end - 1);
}
```

Q18. Convert the following `vec2d`, which models a vector in 2D space with two `doubles` to a class template, `vec2`, which models a vector in 2D space with two `T` values. You only need to modify what's in the given class. Do not implement any extra methods/functions not shown below. Furthermore, if the `operator[]` is called with a value that is not `0` or `1`, an `IndexError` object is thrown as an exception.

Answer:

```
// file: vec2d.h

#ifndef VEC2D_H
#define VEC2D_H

class IndexError
{};

template <typename T>
class vec2
{
  public:
  vec2(T x, T y)
  : x_(x), y_(y)
  {}
  T operator[](int i) const
  {
    if (i != 0 && i != 1) throw IndexError();
    return (i == 0 ? x_ : y_);
  }
  T & operator[](int i)
  {
    if (i != 0 && i != 1) throw IndexError();
    return (i == 0 ? x_ : y_);
  }
  private:
  T x_;
  T y_;
};

#endif
```

Q19. The class below allows me to do this:

```
Square square;
std::cout << square.compute(5) << '\n'; // prints 25
```

Modify the only method in the class so that I get a functor, i.e., so that I can do this instead:

```
Square square;
std::cout << square(5) << '\n'; // prints 25
```

Answer:

```
// file: square.h

#ifndef SQUARE_H
#define SQUARE_H

class Square
{
  int operator()(int x)
  {
    return x * x;
  }
};

#endif
```

Q20. The following will print `"A::f()\n"` twice. Modify it (add one C++ keyword) so that one of the outputs is `"A::f()\n"` and the other is `"B::f()\n"`.

Answer:

```
#include <iostream>

class A
{
public:
    virtual void f() const
    {
        std::cout << "A::f()\n";
    }
};

class B: public A
{
public:
    virtual void f() const
    {
        std::cout << "B::f()\n";
    }
};

int main()
{
    B b1;
    A a1 = b1;
    a1.f();

    B b2;
    A * a2 = &b2;
    a2->f();

    return 0;
}
```

Q21. The following code (obviously!) does not work. Correct it so that the output is

```
grrr ... i'm a wumpus
hi ... i'm a hunter
```

Answer:

```
#include <iostream>
#include <vector>

class LivingThing
{
public:
  virtual void print() const = 0;
};

class Wumpus : public LivingThing
{
public:
    virtual void print() const
    {
        std::cout << "grrr ... i'm a wumpus\n";
    }
};

class Hunter : public LivingThing
{
public:
    virtual void print() const
    {
        std::cout << "hi ... i'm a hunter\n";
    }
};

int main()
{
    std::vector< LivingThing > things;
    things.push_back(Wumpus());
    things.push_back(Hunter());

    for (int i = 0; i < 2; ++i)
    {
        things[i].print();
    }

    return 0;

}
```

Q22. Complete the following that prints the values of `x` in ascending order *without* changing the values of `x`. In other words the value of `x[0]` stays the same, the value of `x[1]` stays the same, etc. The array in `main()` is only a test case. Your program must work for any integer array.

```
#include <iostream>

void bubblesort(int ** start, int ** end)
{
  for (int ** i = start; i < end - 1; ++i)
  {
    for (int ** j = start; j < end - 1; ++j)
    {
      if **j > **(j + 1))
      {
        int * t = *j;
        *j = (*j + 1);
        (*j + 1) = t;
      }
    }
  }
}

int main()
{
    const int n = 5;
    int x[n] = {5, 3, 1, 2, 4};

    int * px[n];
    for (int i = 0; i < n; ++i)
    {
        px[i] = &x[i];
    }

    bubblesort(&px[0], &px[n]);

    // The original array x is unchanged.
    for (int *p = &x[0]; p < &x[n]; ++p)
    {
        std::cout << (*p) << ' ';
    }
    std::cout << '\n';

    // Prints the values in x in ascending order.
    for (int ** p = &px[0]; p < &px[n]; ++p)
    {
        std::cout << (**p) << ' ';
    }
    std::cout << '\n';
```

```
    return 0;
}
```

Instructions

In `main.tex` change the email address in

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

to yours. In the bash shell, execute "`make`" to recompile `main.pdf`. Execute "`make v`" to view `main.pdf`. Execute "`make s`" to create `submit.tar.gz` for submission.

For each question, you'll see boxes for you to fill. You write your answers in `main.tex` file. For small boxes, if you see

```
1 + 1 = \answerbox{}.
```

you do this:

```
1 + 1 = \answerbox{2}.
```

`answerbox` will also appear in "true/false" and "multiple-choice" questions.

For longer answers that needs typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
int x;
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?
\begin{answerlong}
\end{answerlong}
```

you can write

```
What is the color of the sky?
\begin{answerlong}
The color of the sky is blue.
\end{answerlong}
```

For students beyond 245: You can put LaTeX commands in `answerbox` and `answerlong`.

A question that begins with "T or F or M" requires you to identify whether it is true or false, or meaningless. "Meaningless" means something's wrong with the statement and it is not well-defined. Something like "$1+_2$" or "$\{2\}^{\{3\}}$" is not well-defined. Therefore a question such as "Is $42 = 1+_2$ true or false?" or "Is $42 = \{2\}^{\{3\}}$ true or false?" does not make sense. "Is $P(42) = \{42\}$ true or false?" is meaningless because $P(X)$ is only defined if X is a set. For "Is $1 + 2 + 3$ true or false?", "$1 + 2 + 3$" is well–defined but as a "numerical expression", not as a "proposition", i.e., it cannot be true or false. Therefore "Is $1 + 2 + 3$ true or false?" is also not a well-defined question.

When writing results of computations, make sure it's simplified. For instance write 2 instead of $1 + 1$. When you write down sets, if the answer is $\{1\}$, I do not want to see $\{1, 1\}$.

When writing a counterexample, always write the simplest.

Here are some examples (see `instructions.tex` for details):

1. T or F or M: $1 + 1 = 2$ ..... T
2. T or F or M: $1 + 1 = 3$ ..... F
3. T or F or M: $1+^2 =$ ..... M
4. $1 + 2 =$ 3
5. Write a C++ statement to declare an integer variable named `x`.
   ```
   int x;
   ```
6. Solve $x^2 - 1 = 0$.

   Since $x^2 - 1 = (x - 1)(x + 1)$, $x^2 - 1 = 0$ implies $(x - 1)(x + 1) = 0$. Therefore $x - 1 = 0$ or $x = -1$. Hence $x = 1$ or $x = -1$.

7. Which is true? C
   (A) $1 + 1 = 0$
   (B) $1 + 1 = 1$
   (C) $1 + 1 = 2$
   (D) $1 + 1 = 3$
   (E) $1 + 1 = 4$