Exam 1 Review and Practice - Fall 2023

Exam Details

- The exam is made up of:
    - Writing code to solve problems.
    - Tracing execution given specified input values.
- There are multiple versions of the exam.
- **Partial credit is available** for every question.
    - Your goal should be to not need it.
    - Focus your solution on correctness of the problem solving method. Clear evidence of your thought process and plan must be included in the code and comments in order to qualify for partial credit.
    - Put down anything you do know even if you can't solve the entire problem. Some partial credit is better than getting no credit.
    - Make your thinking visible so applying partial credit is easier.
- **Not allowed**
    - Any electronic devices (including calculators, phones, smart watches, and computers)
- **Allowed**
    - A writing utensil (e.g. pen/pencil)
    - Scratch Paper (8.5X11 inches)
    - Up to 5 pages of exam aids that you create.
        - Exam aids can be pages up to 8.5X11 inches and can be handwritten or printed on both sides.
        - You can't use exam aids for scratch work during the exam.
            - If you do, then you must submit the exam aid with your exam.
- We will be manually grading the exam, so don't get stressed if it is not perfect since you will get partial credit. We will also ignore minor errors that do not significantly impact the results.

Exam Strategies

- Understanding at a high level will help you code better during an exam. So, as you study, ask yourself...
    - *What is happening?*
        - *High level: What is the goal?*
        - *Code level: What are the effects and consequences of statements, etc.?*
    - *Why is it that way?*

        - *High level: Why are certain design decisions made? Why is it designed that way?*
        - *Code level: What is the code that accomplishes a higher-level goal? Why is it done that way?*
    - *How is it done?*
        - *High level: How do you accomplish a high-level goal?*
        - *Code level: How do you write the code?*
- Study Strategy: Prioritize how you study to maximize your ability to demonstrate concept mastery while not overwhelming yourself.
    - For a solid passing grade focus on class topics, slides, examples, in-class activities, homework, and labwork.
        - Make sure you understand anything coded in homework, labwork, and in-class examples.
            - What general (higher level) principles were illustrated in a labwork/homework?
                - Could you explain it in general? (i.e. what/why rather than how)
                - Be able to solve and write code for similar problems.
    - After you've done all you can do in the "for a solid passing grade" above, then focus on zyBook Challenge Activities and Participation Activities.
        - If a challenge or activity seems fuzzy, review the text for clarification.
        - Key terms are bolded.
    - After you've done all you can from the previous approaches and if you are still fuzzy about a topic, read about it in a different source or textbook. You can also talk to someone who can explain it in a different way.

Exam Topics

Note, topics are not always organized sequentially!

Getting Started

- Compile Process
- Program Flow / Control Structures
- First Program
    - Basic IO
    - Comments & Whitespace
    - Errors and Warnings

Variables / Assignments

- Operators
- Assignment
- Expressions

Overarching Themes

- Software Development Process
    - Design
        - Flowcharts
        - Pseudocode
- Header files
- Coding Style / Readability
- Debugging

Selection

- Selection (e.g. if, switch)
- Logical Operators
- Boolean data type
- Conditional Expressions
- Floating-point comparison

Iteration

- Iteration (e.g. while, for, do while)
- Nested Loops

Strings

- Comparisons
- Access
- Modify
- Character Operation

Compound Data

- Array
    - Declaring / Defining / Initializing
    - Traversing
    - Passing as array parameters
    - Memory Safety
    - C-string
- Parallel Arrays
    - (i.e. Multiple Arrays in zyBook)
- Linear search

Exceptions

- Why do we use them?
- throw
- try
- catch

Functions

- Signature/parameters/arguments (formal vs. actual)
- Call and return
- Scope
- Declarations vs definitions
- Unit Testing

Practice Problems

Note that practice problems are generally **greater than or equal in difficulty** to what you will find on the exam. Exam questions are designed to be challenging and doable in the time provided. Some of the practice questions are longer than what you would see on an exam, but should give you good practice.

Additional practice problems can be doing the parts of the labworks that you did not work on.

Multiples

Given positive integers `n`, `k` and `l`, print the first `n` positive integer numbers that are multiple of `k`, `l` or both. Example: `n = 6`, `k = 2` and `l = 3`, you should print:

```
2 3 4 6 8 9
```

Triproduct numbers

A positive integer number n is triproduct if it can be obtained by the product of three consecutive positive integers. For example, 120 is triproduct, since `4 * 5 * 6 = 120`. Given `n > 0`, determine if `n` is triproduct.

Array Segments

Given `n > 0` and a sequence of `n` integer numbers, print how many segments composed by consecutive copies of the same number the sequence has.

For example: The sequence

```
5 2 2 3 4 4 4 4 1 1 1
```

has 5 segments.

Find the sum

Write a program that given:

- `n` where `0 < n < 5000`
- a list of n numbers in increasing order
- a target number `k`

prints two distinct elements from the list such that their sum is equal to the target `k` if such pair exists and ”none” otherwise.

Examples:

- For `n = 3`, list `2 4 6` and `k = 13`, the output is `none`.
- For `n = 8`, list `1 2 3 4 5 6 7 8` and `k = 8`, valid outputs are:
    - `1 7`
    - `2 6`
    - `3 5`

Review continues on the next page.

Numeric Palindrome

Write a boolean function that takes an integer parameter and returns true if the base 10 number is a numeric palindrome. Note, you cannot use any type of string to solve this problem.

`bool isNumericPalindrome(int n)`

For example

- `isNumericPalindrome(121) → true`
- `isNumericPalindrome(-121) → true`
- `isNumericPalindrome(220) → false`
- `isNumericPalindrome(-220) → false`
- `isNumericPalindrome(18344381) → true`
- `isNumericPalindrome(12345) → false`
- `isNumericPalindrome(1) → true`
- `isNumericPalindrome(0) → true`

Happy numbers

You should do this without converting numbers to strings and vice versa.

A happy number is a non-negative integer that eventually becomes 1 when iterated over the sum of squared digits function.

For example, 28 is happy:

$$28 \rightarrow 2^2 + 8^2 = 68 \rightarrow 6^2 + 8^2 = 100 \rightarrow 1^2 + 0^2 + 0^2 = 1 \checkmark$$

But 4 is unhappy (omitting intermediate results), as the chain of numbers led to the

original number.

$$4 \rightarrow 16 \rightarrow 37 \rightarrow 58 \rightarrow 89 \rightarrow 145 \rightarrow 42 \rightarrow 20 \rightarrow 4 \ldots$$

In fact, every unhappy number eventually converges with 4.

Write a function bool `isHappy(int n)` that receives as an argument a non-negative integer and returns true if the number is happy.

Examples:

- `isHappy(4)` returns `false`.
- `isHappy(13)` returns `true`.
- `isHappy(28)` returns `true`.

Review continues on the next page.

Pesky Bug

Your friend has written a function that looks to see if there is a failing grade in a list of grades in an array. The code compiles and runs but both g1 and g2 state that there are no failing grades when g2 clearly has a 55 which is a failing grade. They thought it might be a C++ thing, but equivalent implementations in Python and Java gave the same results!

What is the logic problem with your friend's code?

A failing grade is anything less than 60.

```
bool hasFailingGrade(double grades[], unsigned int size) {
  if (size == 0) {
    throw invalid_argument("There are no grades.");
  }

  bool hasFailing = false;
  for (unsigned int i = 0; i < size; ++i) {
    if (grades[i] < 60) {
      hasFailing = true;
    }
    else {
      hasFailing = false;
    }
  }
  return hasFailing;
}
```

If you pass in

- `double g1[] = {83, 91, 100, 60, 75}; // no failing grades`
  - Returns false as expected
- `double g2[] = {100, 100, 55, 100};   // has a failing grade`
  - Returns false when it should return true

Review continues on the next page.

Average Donation

Sarah and Kelly are devoted to donating to several different causes. Both want to receive the prize for the “number 1 donor of the department” they work in. The rules for the prize specify that the winner will be the donor with the maximum average donation, as long as some conditions are satisfied:

1. the winner must have donated to at least 5 different causes;
2. the total amount donated is at least 300 dollars.

If the average is the same, the winner is the one with most donations. Sarah and Kelly want to compare their donations to determine who among them would win the prize (if any). They each take note of their total donation to each cause (as a non-negative integer value) and signify the end of the list with a negative value.

- For the donations below, Sarah wins

Sarah	100	200	50	100	100	51	-10
Kelly	100	100	100	100	100	100	100	100	-40

- For the donations below, Kelly Wins

Sarah	500	700	-100			
Kelly	50	100	50	50	50	-100

- For the donations below, no one qualifies for the prize

Sarah	10	200	50	10	10	5	-10
Kelly	1000	-40					

- For the donations below, they tie

Sarah	100	200	50	100	100	50	-10
Kelly	10	130	50	150	70	190	-40

Review continues on the next page.

Your task in this problem is to write a function

```
void bestDonor(int kellyDonations[], int numKellyDonations,
               int sarahDonations[], int numSarahDonations)
```

that takes as input their donations and output the outcome to standard out: either Sarah wins, Kelly wins, they tie, or they do not qualify for the prize.
(A function declared as returning 'void' does not return any specific value)
Notice that the arrays have a sentinel value (a negative integer at the end), so you can solve this problem without the size argument:

```
void bestDonor(int kellyDonations[], int sarahDonations[])
```

Words of Importance

A former president wanted to make their mark by transforming the way we determine the importance of words by emphasizing vowels including 'y'!

When comparing two words, each word is given a weight.

- A word receives three points every time a vowel is compared to a consonant.
- A word receives one point when comparing two vowels or two consonants if its character is greater than the other in the traditional sense.
- If one word has a vowel and the other has no character, then the word gets two points.
- If one word has a consonant and the other has no character, then one point is added.
- If both words have the same character, then no points are assigned.

The word with the most points is more important.

Write a function that takes in two C-Strings and returns 1 if the first word is more important than the second and returns 2 if the second word is more important and returns 0 otherwise.

Examples

- moreImportant("eel", "the") returns 1.
- moreImportnat("the", "eel") returns 2.
- moreImportant("angelic", "mistrust") returns 2.
- moreImportant("spy", "il"') returns 1.
- moreImportant("ill", "spy") returns 2.
- moreImportant("trust", "silly") returns 2.
- moreImportant("see", "sea") returns 1.

Note: this is case insensitive comparison. The toupper() and tolower() functions can help.

Review continues on the next page.

Some Number Theory Problems That You Could Brute Force

You should do this without converting numbers to strings and vice versa.

1. Which digit is missing from 2^{29}?
2. Consider the following triangle of numbers:

$$
\begin{array}{c}
1^2 \\
2^2 \quad 3^2 \\
4^2 \quad 5^2 \quad 6^2 \\
7^2 \quad 8^2 \quad 9^2 \quad 10^2 \\
\cdots
\end{array}
$$

   What is the sum of the n^{th} row? (assume n sufficiently small that the sum will not overflow an unsigned long)
3. Find all positive integers 4 <= x, y, z <= 40 such that
   a. x + y + z = 62
   b. x * y * z = 2880
4. Find all non-negative integers 0 <= a, b, n < 32 such that $2^a + 2^b = n!$.
5. How many 4-digit numbers abcd, where digits a,b,c,d are distinct, satisfy “abcd + dcba is a multiple of 101”?
   a. E.g. 1234 satisfies the property because all digits are distinct and 1234 + 4321 = 5555 = 55*101.
6. How many pairs (m,n) of positive integers 1 <= m,n <= 1000 satisfy “$m^2 + n^2$ is divisible by 121“?

Automate Your Calculus and Physics Homework

1. Write a program that takes as input a polynomial and outputs the derivative of the polynomial. The derivative f’(x) of f(x) = g(x) + h(x) is f’(x) = g’(x) + h’(x). The derivative of a polynomial term ax^b is (a*b)x^(b-1), except for the special case of b=0 which has derivative 0.
   a. E.g. input: “x^2 + 2x + 1”; output: “2x+ 2”
   b. Easy mode: assume the degree is at most N, for some constant integer N.
   c. Medium mode: Easy mode + handle negative exponents
      i. E.g. input: “x^-2”; output: ”-2x^-3”
   d. Challenge mode: Medium mode without assumption on N (i.e. handle arbitrarily large N)
      i. E.g. input: “x + x^3 + x^2007 + x^-1200”; output: “1 + 3x^2 + 2007x^2006 - 1200x^-1201”
2. Write a program that takes as input a polynomial and outputs the indefinite integral of the polynomial.
   a. E.g. input: “x^2 + 2x + 1”; output: “(1/3)x^3 + x^2 + x + c”
3. Write a program that solves all variations of the following kinematics problem: “A block of mass <m> kg is held at a height of <h> m on an inclined plane at angle <a> degrees. The block is initially at rest. The coefficient of sliding friction between the block and the plane and the table (the plane and table are the same material) is <k>. At time t=0 seconds, the block is released and begins to slide down the inclined plane. How far in the horizontal direction does the block travel?”
   a. E.g input: m = 10, h = 1, a = 60, k =-0.3; output: “1.26635 m”
   b. You need to know:

i. $F = ma$
    1. F is the force on the object,
    2. m is the mass of the object, and
    3. a is the acceleration of the object.
ii. For static equilibrium, all forces sum to 0.
iii. For kinematics, such as this problem, there is some net force leftover and therefore the object will accelerate in the direction of that force.
iv. The acceleration due to gravity is 9.81 m/s^2.
v. The force due to sliding friction acts opposite to the direction of motion and has magnitude kF_N
    1. k is coefficient of sliding friction and
    2. F_N is the normal force which acts perpendicular to the plane on which the object slides.
    3. The normal force is the equal and opposite reaction, the upward pushing force that, for example, the Earth exerts on your feet so that you can stand on the Earth rather than being pushed into the Earth.
vi. Acceleration is the rate of change of velocity, i.e. the derivative of velocity.
    1. $a = dv/dt$
vii. Velocity is the rate of change of position (or displacement), i.e. the derivative of position.
    1. $v = dx/dt$
viii. The big 4 kinematics equations are:
d := displacement
v_i := initial velocity
v_f := final velocity
a := acceleration
t := time
    1. $d = 1/2at^2 + v_0t$
    2. $v_f^2 = v_i^2 + 2ad$
    3. $v_f = v_i + at$
    4. $d = (v_i + v_f)t/2$
ix. In this problem, the block gains speed (due to gravity) as it slides down the inclined plane. That acceleration is limited by the resistance due to friction. Once the block reaches the table, only friction is acting on the block (in the horizontal direction; gravity still acts vertically but the block is in static equilibrium with respect to the vertical axis) so the block slows down and eventually stops. The force due to sliding friction goes away as soon as the block stops sliding (i.e. it does not start moving the block back towards the inclined plane). You can assume that sliding friction engages and disengages instantaneously. You can assume that the transition from the inclined plane to the table happens instantly (i.e. the block is a point mass and is at the end of the inclined plane at one instant and on the table at the next, with no time in between and no displacement during the transition (no time = no motion).