ECE 650 - Midterm Exam

Below find the midterm exam instructions.

* Please read all instructions carefully.
* There are 3 questions on this exam, all with multiple parts. This exam will be graded out of a maximum of 100 points.
* Please submit all solutions via the appropriate dropbox on LEARN. No other form of submission shall be accepted. We prefer that you submit your solutions exactly once.
* You have 2 hours and 30 minutes to complete the exam. If you are not able to submit within the time allotted, you will get 0 points.
* The exam is open book and notes. You are allowed to consult resources on the internet. However, you are not allowed to simply copy solutions from any source. Instead, your solutions must be in your own terms. Further, you may not consult with each other or any other person, other than the professor and/or TAs. You can reach us via Piazza or email.
* **For each question below, please write your answers in the space provided right after the question.** Please make sure to write your names and IDs on the first page of this exam book.

## You may submit your solutions as a PDF file, generated using MS-Word or Latex editor or some other suitable editor. We also provide an editable version of the exam via LEARN. We discourage hand-written solutions. If you insist on hand-written solutions, then please make sure your handwriting is legible.

* Solutions will be graded on correctness, clarity, completeness and brevity. Most problems have a relatively simple and straightforward solution. We may choose to give some credit for partial solutions.
* Your completed solutions are due at 11:32 AM ET on Saturday Oct 30, 2021. Submit via the appropriate dropbox on LEARN.

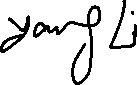
NAME: Yang Li

Email ID: y3372li@uwaterloo.ca

Student ID: 20974262

In accordance with the letter and the spirit of the University of Waterloo honor code, I pledge that I will neither give nor receive assistance on this examination.

SIGNATURE:



|  |  |  |
| --- | --- | --- |
| Problem | Max points | Points Received |
| Q1 | 30 |  |
| Q2 | 40 |  |
| Q3 | 30 |  |
| Total | 100 |  |

# Question 1: Regular Expressions (30 points)

Each of the questions below are worth 7.5 points. Please provide a pithy, precise, correct and complete answer. You only need basic regular expression (regex) operations to answer these questions. The alphabet for all the languages listed below is *{*0*,* 1*}*, unless specified otherwise.

1. **Language to Regex:** Write a regular expression for the following language *L* presented in set theoretic notation. No need to justify your answer.

## Solution:

*L* : *{w | w* starts and ends with a different character from the alphabet*}*

*R: 0(0+1)\*1+1(0+1)\*0*

1. **Equivalence of Regexes:** Is the following sentence S true? If your answer is YES, provide a justification. Else, provide a string that belongs to one of the two regular expressions below, but not in the other.

S: The following two regular expressions represent the same language over the alphabet *{*0*,* 1*}*:

*r*1 = 0(0 + 1)∗0 + 00(0 + 1)∗11 + 11(0 + 1)∗00 + 11(0 + 1)∗11

*r*2 = (00 + 11)(0 + 1)∗(00 + 11)

## Solution:

## False.

## Consider the string ‘0110’, This string is clearly part of the language characterized by the part of regular expression r1 0(0+1)\*0 , however is not part of the language described by the regex r2. The reason is that, any string accepted by r2 must start with ’00’ or end with ’11’. Unfortunately, the string ’0110’ does not satisfy either of these properties.

1. **Equivalence of Regexes:** Is the following sentence S true? If your answer is YES, provide a informal, yet precise, justification. Else, provide a string that belongs to one of the two regular expressions below, but not in the other.

**Sentence S:** The following two regular expressions represent the same language over the alphabet Σ = *{a, b}*:

*r*1 = (*ab*∗)∗ + (*a*∗*b*)∗ + (*a*∗*b*∗)∗

*r*2 = (*a* + *b*)∗

## Solution:

True.

Both of these regular expression accepts the language for strings: the universal language over the alphabet {a, b}.

1. **Properties of Regex:** Is the sentence S below true? If your answer is YES, then provide your reasoning as to why the sentence is true. If NO, provide a counterexample.

**Sentence S:** If a language represented by a regular expression R is infinite, then the regular expression R must contain a Kleene star.

**Solution:**

NO. Here is a counter example:

λ∗

# Question 2: programming language concepts (40 points)

For each of the sub-questions below, provide a concise, correct, and complete answer. Each sub-question is worth a maximum of 10 points.

1. **Basic Object-Oriented Design and Data Encapsulation:** After learning about basic OO design and data encapsulation, Ian realized that his code below in listing 1 violates the principle of data encapsulation. He then went ahead and made a very minor chage to the getter function get a() to obtain the code in listing 2:

1

class Base { private:

int a; public:

Base () {...}

int\* get\_a () {

...

return & a;

}

};

2

3

4

5

6

7

8

9

10

Listing 1: original code

1

class Base { private:

int a; public:

Base () {...}

int& get\_a () {

...

return a;

}

};

2

3

4

5

6

7

8

9

10

Listing 2: modified code

Is Ian’s modified code in listing 2 fundamentally different from the original code in listing 1, when viewed from the point of view of violation of the principle of data encapsulation?

## Solution:

NO, I think they are similar. In list 2, the getter function get a() returns a reference to the private data in the class Base. This completely violates the idea of data encapsulation because now any external code can access and manipulate this data in a object of class Base. If he wants to use pointer or reference here, I suggest him setting the variable **const.**

1. **Templates in C++:**In class we studied function and class templates. Chunxiao recently wrote the following two pieces of code, both using function templates. He was very excited when the first piece of code worked correctly, but the second one gave a compile error. From our discussion of the sample midterm, he realized that he had forgotten to overload the *<* operator for the ADT class and hence he was getting a compile error. Hence, he went ahead and fixed that issue by overloading the *<* operator. However, he continued to get a compile error. Can you find the issue for him and tell him how he can fix it?

1

template < typename T>

T min( T a, T b) { return a < b++ ? a : b;}

int main (){

int x;

x = min <int >(10 , 3); return x;

}

2

3

4

5

6

7

8

Listing 3: code using templates

1

template < typename T>

T min( T a, T b) { return a < b++ ? a : b;}

class ADT {// Abstract DataType private:

int value; public:

ADT ( int v) : value( v) {}

friend bool operator < ( const ADT & t1 , const ADT & t2 ) { return t1 . v < t2 . v;

}

};

int main (){

ADT x(10) , y(4);

y = min <ADT >(10 , x);

return y;

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Listing 4: code using templates and an ADT

## Solution:

## There are 3 problems here.

## First, in the class ADT, we only have one variable named ‘value’. But in line 9, for a instantiate ADT, t1 and t2, it’s obvious that they don’t have a variable called ’v’. So here we should change this line into: { return t1.value < t2.value;}

## Second, in the main function, it should return a integer. But now we return a class in line 16. So here we should define a get function named ‘get\_value’ which will return the value of ADT as a integer. in the class as public, and return y.get\_value() in the main function.

## Third, we only define a ‘<’ operator for a class. However, in the template, there is another operator’++’. So we should define it in the class.

1. **Semantics of Inheritance and Constructors in C++:** Mary hastily wrote the following code (listing 5) and got a very unexpected compile error. She quickly realized her error and went ahead and fixed it in the modified listing 6 below. However, now she is not able to instantiate objects of the base class. What is going on?

1

# include <iostream > class base {

private:

base () {

std :: cout << " Hey: I am the base constructor ()\ n";

}

};

class derived : public base { public:

derived () {

std :: cout << " Listen , I may be derived , but I am better than you\ n";

}

};

int main () { derived d;

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

Listing 5: original code

1

# include <iostream > class base {

protected :

base () {

std :: cout << " Hey: I am the base constructor ()\ n";

}

};

class derived : public base { public:

derived () {

std :: cout << " Listen , I may be derived , but I am better than you\ n";

}

};

...

2

3

4

5

6

7

8

9

10

11

12

13

14

Listing 6: modified code

## Solution:

Different from other function, the constructor cannot be inherited by the derived class. Therefore, in order to initialize the base class, she needs to call the constructor of the base class in the derived class (namely, an explicit call).

1. **Just-in-Time Compilers:** Describe the purpose of Just-in-Time compilers, i.e., in what context are they used and why?

**Solution:**

The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance by compiling bytecode into native machine code at runtime.

At a language runtime, take Java as an example, the JVM loads the class files, determines the semantics of each individual bytecode, and performs the corresponding calculations. The additional use of processors and memory during interpretation means that Java applications will execute slower than native applications. The JIT compiler helps improve the performance of Java programs by compiling bytecode into native code at runtime.

This is in contrast to a traditional compiler that compiles all the code to machine language before the program is first run. So JIT is suitable for dynamic language.

# Question 3: Processes, System Calls/Signals, Control-hijack Attacks, Compiled vs. Interpreted Languages (30 points)

Each of the questions below are worth 10 points. Please provide a pithy, precise, correct and complete answer for each sub-question.

1. **Overflow:** Is there a possibility of an integer and a buffer overflow in the code below. If there is an integer overflow, can it cause a buffer overflow in the code below. Explain your answer in detail. (Note: Not all integer overflow errors may cause a buffer overflow. And we assume int type is 32 bits.)

1

...

int nresp = packet\_get\_int (); // read nresp number of bytes from input stream if ( nresp > 0) {

if(0 x0 == nresp \* sizeof( char \*)) {

std :: cout << " Integer overflow detected \ n"; exit (-1);

}

response = malloc( nresp \* sizeof( char \*));

for ( i = 0; i < nresp ; i++) // read input stream response[ i] = packet\_get\_string (...);

}

...

}

2

3

4

5

6

7

8

9

10

11

12

13

Listing 7: Code Fragment from OpenSSH

The precise and detailed semantics of packet get int() and packet get string(...) are irrele- vant to answering the question. The only pieces of information you need are the following, namely, that packet get int() returns an 32-bit integer, and that packet get string(...) returns a string controlled by the attacker.

## Solution:

If the variable nresp has the value like 0xe0000000, then nresp\*sizeof(char\*) is guaranteed to have an integer overflow. In C++, malloc receive the variable as type size\_t, and size\_t is 64 bit in a 64 system. However, int is 32 bit, which means that malloc is being asked to allocate a extremely big memory. But maybe it won’t cause a buffer overflow.

1. **Process Memory Layout and Call Stack:** The semantics of the call stack, as described in class, is typically referred to as Control-Flow Integrity (CFI). In other words, we say that the CFI of a program P is preserved if the code of P is executed in the sequence it is laid out by the programmer from top to bottom, say, as given in the main() function. Otherwise, we say that the program’s control-flow integrity has been violated.

Are there ways in which someone can violate the CFI of a running program?

(Hint: To get full points describe how the call stack of a typical process works during its normal execution such that its CFI is preserved, i.e., the control-flow of the program follows sequentially in the order in which the code is specified by you. Then describe a method by which the CFI of a running program can be violated.)

## Solution:

The things that are stacked after the LIFO principle are not the local variables but the entire stack frames ("calls") of the functions being called. The local variables are pushed and popped together with those frames in the so-called function prologue and epilogue, respectively.

Inside the frame the order of the variables is completely unspecified; Compilers "reorder" the positions of local variables inside a frame appropriately to optimize their alignment so the processor can fetch them as quickly as possible. The crucial fact is that the offset of the variables relative to some fixed address is constant throughout the lifetime of the frame - so it suffices to take an anchor address, say, the address of the frame itself, and work with offsets of that address to the variables. Such an anchor address is actually contained in the so-called base or frame pointer which is stored in the EBP register. The offsets, on the other hand, are clearly known at compile time and are therefore hardcoded into the machine code.

So shortly said, the code just accesses them directly via constant compile-time offsets from the base pointer; It's simple pointer arithmetic.

1. **ROP Attacks and ASLR:** In class we studied Address Space Layout Randomization (ASLR), a really interesting and clever defense mechanism against Return-Oriented Programming (ROP) attacks. What is a key difference between basic ROP attack and buffer overflow-based code injection attack on the stack? Could a canary-based defense mechanism be effective against ROP attack? Justify your answer.

**Solution:**

Buffer overflow refers to incorrect boundary checking or processing implicit length data (such as strcpy or strcat) to allow malicious input to write memory to the end of the array.

In addition to the new return address, your malicious data will also contain more data, which will be located in the memory below and above the return address. Part of it is the payload. A ROP attack is a payload that can be provided to a buffer on the stack through a buffer overflow vulnerability. (Overflowing other buffers allows you to overwrite other data, such as in structures or other global variables nearby, but it cannot control the program counter.)

No, it can’t.

Stack Canaries work by modifying the prologue and epilogue areas of each function to place and check values on the stack respectively. In this way, if the stack buffer is overwritten during the memory copy operation, the execution will return from the copy function before the error is noticed. When this happens, an exception will be thrown, and the exception will be passed up back to the exception handler hierarchy until it finally reaches the default exception handler of the operating system. If you can overwrite the existing exception handler structure in the stack, you can point it to your own code. This is a structured exception handling (SEH) vulnerability that allows you to skip the Canary check altogether.