University of Toronto

Duration: 2 hour 30 minutes
Examiner: Kuei (Jack) Sun

Please fill your student number, last and first name below and then read the instructions carefully.

Student Number: ____ ____ ____ ____ ____ ____ ____ ____ ____ ____

Last Name: _____

First Name: _____

Instructions

Examination Aids: Examiner approved aid sheet is allowed.

Do not turn this page until you have received the signal to start.

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted. Use the blank space in last page as scratch space. Its content will not be marked.

This exam consists of 8 questions on 20 pages (including this page). The value of each part of each question is indicated. The total value of all questions is 150 marks.

For the written answers, explain your reasoning clearly. Be as brief and specific as possible. Clear, concise answers will be given higher marks than vague, wordy answers. Marks will be deducted for incorrect statements in an answer. Please write legibly!

Work independently.

MARKING GUIDE

Q1:	_____	(11)
Q2:	_____	(23)
Q3:	_____	(17)
Q4:	_____	(28)
Q5:	_____	(10)
Q6:	_____	(23)
Q7:	_____	(26)
Q8:	_____	(12)
Bonus:	_____	(4)
Total:	_____	(150)

Question 1. True or False [11 marks]

Circle **T** if the statement is true, otherwise circle **F** if the statement is false. 1 mark each.

1. In Python, `yield` is a statement. **T** **(F)**

2. Python closures always capture outer variables by reference. **(T)** **F**

3. C++ disallows overloading the dereference operator. **T** **(F)**

4. SFINAE enables reflective programming in C++. **(T)** **F**

5. Programming languages that perform late binding must have a method resolution order. **(T)** **(F)**
   This question is under-specified. Should also say "that supports inheritance".

6. Suppose `foo` is an immutable container that stores integers and overloads the subscript operator, then `foo[0] = 5` is a semantically legal operation. **T** **(F)**

7. The purpose of generic programming is code reuse across different data types. **(T)** **F**

8. Rust allows implicit widening conversion. **T** **(F)**

9. The Rust compiler will attempt to copy an object *only after* it decides that moving the object is not permissible. **T** **(F)**

10. The Rust compiler can automatically detect and prevent deadlocks from occurring. **T** **(F)**

11. The `map` function in Python is an example of lazy evaluation. **(T)** **F**

Question 2. **Multiple Choices** [23 marks]

Pick all answer(s) that are correct. You will lose 1 mark per wrong choice, down to 0 marks.

a) For each programming language feature, circle whether C++11, Python, or Rust supports it (more than one choice is possible). [7 marks]

i. Default Parameter

C++11 **Python** Rust

ii. Function Overloading

C++11 Python Rust

iii. Closure

C++11 **Python** **Rust**

iv. Return Type Polymorphism

C++11 *Python* **Rust**

*no marks lost or gained on this choice

v. Tuple Unpacking

C++11 **Python** **Rust**

b) Which of the following are higher order functions? [4 marks]

i. `sum`

ii. `filter`

iii. `decltype`

iv. `constexpr` functions

v. Python decorators

c) Which of the following can cause the fragile base class problem to occur? [4 marks]

i. Adding the `final` specifier to the base class. (e.g. `class Foo final`)

ii. Adding a non-virtual function to the base class.

iii. Adding a pure virtual function to the base class.

iv. Overriding a virtual function in the derived class.

v. Adding new member variables to the derived class.

d) Which of the following are valid C++ assignments? Note: `i` and `j` are global variables. [4 marks]

```
int i=2, j=3;        int & r = j;        int & foo() { return i; }
```

i. `r = i + j;`

ii. `(foo() ? i : r) = 9;`

iii. `i = *r;`

iv. `r = foo();`

v. `foo() = 5;`

e) Which of the following describes Rust's type system? [4 marks]

i. Inferred typing

ii. Strongly typed

iii. Duck typing

iv. Covariant containers

v. Dynamically typed

Question 3. **Short Answers** [17 marks]

a) List two properties that C3 Linearization guarantees. You must explain what each guarantee means to receive full marks (e.g. with an example). [4 marks]

Preserves local precedence order – i.e. the order in which the parent classes are inherited

Preserves monotonicity – linearization of a class will not change order regardless of the inheritance hierarchy it may be in

[2 marks per correct statement and explanation of the terms]

b) Describe two similarities and two differences between Python mixins and Rust traits. [4 marks]

Similarities: [2 marks – 1 mark per correct statement]

Both can supply default implementation.
A structure/class can have multiple mixins or multiple traits.
Both can be used for code reuse.

Differences: [2 marks – 1 mark per correct statement]

Traits cannot have data fields, mixins can.
Traits are not affected by order of inheritance, mixins can be.
Traits are supposed to be implemented (is-a), mixins are supposed to be included (has-a).
Traits provides an interface, mixins only provides implementation.

c) Expand the C preprocessor macro function, MYSTERY. [9 marks]

```
#define S(s) #s                          #define v hello
#define C(t, a...) t ## a                #define m world
#define X(t, a...) C(_ ## t, a)
#define I(f, a) f a
#define P(p) (p, printf)

#define MYSTERY(m) X(v, I(C, P(s)))(v, S(v) S(m), m)

MYSTERY(m);    // expand this
```

```
MYSTERY(m)

MYSTERY(world)   // no direct # or ## in MYSTERY so must
                 // expand argument first

X(v, I(C, P(s)))(v, S(v) S(world), world)   // cannot expand v because
                                            // of ##, can expand I

X(v, I(C, (s, printf)))(v, S(v) S(world), world)   // expanded P

X(v, C (s, printf)) (v, S(v) S(world), world)      // expanded I

X(v, sprint) (v, S(v) S(world), world)             // expanded C

C(_v, sprintf)(v, S(v) S(world), world)            // expanded X

_vsprintf(v, S(v) S(world), world)                 // expanded C

_vsprintf(hello, S(v) S(world), world)        // replaced v

_vsprintf(hello, "v" S(world), world)         // expanded S
                                              // Cannot replace v inside S(v)
                                              // because of #
_vsprintf(hello, "v" "world", world)
```

4 marks for expanding X(v, I(C, P(s))) correctly

4 marks for expanding each term of (v, S(v) S(m), m) correctly

Question 4. **Object-Oriented Programming** [28 marks]

```
struct X {
    char x[15];
    virtual void foo() {
        print("X.foo");
    }
    void bar() {
        print("X.bar");
    }
};

struct Y : X {
    int y;
    void bar() {
        print("Y.bar");
    }
    void baz() {
        print("Y.baz");
    }
};
```

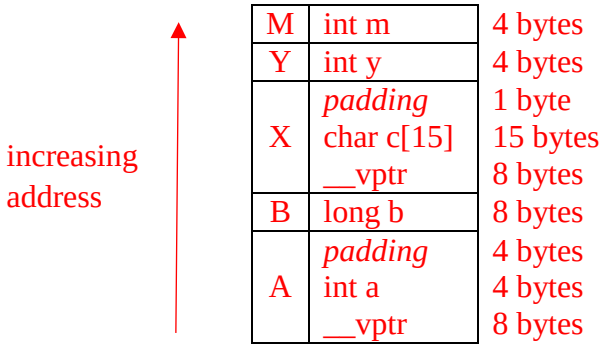
```
struct A {
    int a;
    virtual void qux()=0;
    void baz() {
        print("A.baz");
    }
};

struct B : A {
    long y;
    virtual void tor() {
        print("B.tor");
    }
    void bar() {
        print("B.bar");
    }
};
```

```
struct M : B, Y  {
    int m;
    virtual void qux() { print("M.qux"); }
    virtual void tor() { print("M.tor"); }
};

int main() {
    M mp;
    Y & yp = mp;
    X & xp = yp;
    B & bp = mp;
    A & ap = mp;

    /* method calls for part d. */
    return 0;
}
```

a) Suppose the address of `mp` is `0x7ffd69d24c90`, what is the address of `yp`? [1 mark]

`0x7ffd69d24ca8`

b) Draw the data structure layout for class M. Assume 8-byte alignment and the target architecture is 64-bit. Draw a line to show increasing address. [9 marks]



1 mark for each correctly placed variable/padding/vptr.

-1 mark per incorrectly placed class

-1 mark for extra padding/vptr

c) For the inheritance hierarchy shown above, Write the entries in each of the five classes' virtual table in the correct order. Also state of the size of class M in bytes. [9 marks]

`X::vtable`

```
X::foo
```

`A::vtable`

```
nullptr
```

`Y::vtable`

```
X::foo
```

`B::vtable`

```
nullptr
B::tor
```

`M::vtable`

```
X::foo
M::qux
M::tor
```

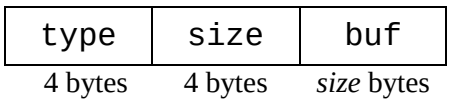
`sizeof(M) = 56`

d) What is the output for the following method calls? If the method call is illegal (either a runtime or compile-time error), state the issue. [9 marks]

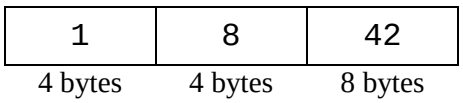
`xp.foo()`	X.foo
`xp.bar()`	X.bar
`xp.baz()`	Illegal – class X does not have method named baz
`ap.qux()`	M.qux
`ap.tor()`	Illegal – class A does not have method named tor
`bp.tor()`	M.tor
`bp.baz()`	A.baz
`mp.foo()`	X.foo
`mp.bar()`	Illegal – bar is ambiguous between X.bar and B.bar

Question 5. Template Metaprogramming [10 marks]

In assignment 3, one of the tasks involved was building the client to send request packets to the server. Recall that the format of Value looks like this in a packet:



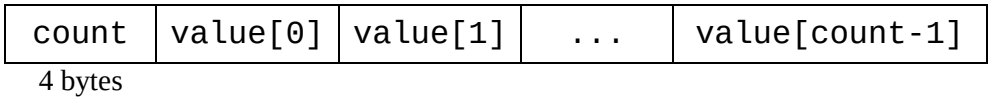
Integer Example:



Where type is one of:

```
enum ValueType { INTEGER = 1, FLOAT = 2, STRING  = 3, };
```

And that a Row contains an array of values like this:




```
struct Request {
    Packet packet;
    void putvalue(long val);
    void putvalue(double val);
    void putvalue(const char *);

    template<typename... Args>
    void putrow(Args&&... args);
} req;
```

```
class Packet {
    /* internal buffer */
public:
    //  use this function to
    //  add data to the end
    //  of the buffer
    void pack(const char * bytes,
              int size);
};
```

a) Complete the implementation for serializing a 64-bit integer into a packet. You do not have to perform endianness conversion. The `type` field needs to be `INTEGER`, and the `size` field needs to `sizeof(long)`. Hint: use `packet.pack` to serialize data into the buffer. [0 marks]

```
void Request::putvalue(long val) {

    int temp = INTEGER;
    packet.pack((const char *)&temp, sizeof(int));
    temp = sizeof(long);
    packet.pack((const char *)&temp, sizeof(int));
    packet.pack((const char *)&val, temp);



    /* 1 mark for each line of code */
    /* THIS QUESTION WAS REMOVED FROM FINAL EXAM */
    /* DO NOT MARK */
}
```

b) Given the following sample usage:

```
req.putrow(123, 32.5, "hello world");  // packs 3 values, count = 3
long a = 234; const char * msg = "good luck";
req.putrow(a, msg);                    // packs 2 values, count = 2
```

Implement `putrow` for Request so that it is possible to pack any combination of integers, floats, and c-strings. Assume `putvalue` for int, double and c-string have also been implemented. Note: you are *required* to use move semantics for this question. [10 marks]

```
struct Request {
    /*
     * other declarations. Add your own helper methods if needed
     */

    template<typename... Args>
    void putrow(Args&&... args) {  // remember to close braces

        int temp = (int)sizeof...(Args);

        packet.pack((const char *)&temp, sizeof(int));

        putvalues(std::forward<Args>(args)...);
    }

    void putvalues() const {}

    template<class T, typename... Args>

    void putvalues(T&& val, Args&&... args) {

        putvalue(std::move(val));

        putvalues(std::forward<Args>(args)...);
    }

    /* 1 mark for each line [8 marks] */
    /* 1 mark for std::move and T&& val [1 mark] */
    /* 1 mark for std::forward [1 mark] */
    /* you get 8 out of 10 if you don't use move/forward and/or && */




}; // struct Request
```

Question 6. **Reflective Programming** [23 marks]

You are developing a package for adding type safety checks to the end users' custom-defined classes. The interface you provide requires that their classes inherit from `Base`, and that the fields they want to type check be specified as a class member attribute of type `Field`, such as:

```
class User(Base):
    name = Field(type=str)
    age = Field(type=int)
    height = Field(type=float)

fred = User(name="Fred", age=23, height=6.2)
anne = User(name="Anne", age="19") # error: age is not an integer
```

a) Complete the class `Base`, with metaclass named `Meta`, such that the `__init__` method of `Base` takes in a variable number of keyword arguments and stores each key value pair as a field in `Base`, with the field name of each pair being its key. You may not make use of the instance's `__dict__` attribute for this question. [4 marks]

```
class Base(metaclass=Meta):
    def __init__(self, **kwargs):
        for col, val in kwargs.items():
            setattr(self, col, val)
```

1 mark for each line of code

-1 mark if metaclass specified as super class

b) Write a descriptor class named `Field` such that upon intercepting attribute assignment, it checks that the type of the value matches what is specified in the constructor. If a mismatch occurs, raise an `AttributeError` with the message "wrong type". [*continue to next page*]

[*continue from last page*] Next, write a metaclass, `Meta`, to work the `Field` class such that user-defined classes support multiple instances correctly. Note that field order does not matter. Hint: how can you use the metaclass so that each `Field` instance knows its attribute name? [13 marks]

```
class Field:
    def __init__(self, type):
        self.type = type

    # complete class Field here, and add metaclass Meta

    # 2 marks (1 for each line)
    def setname(self, name):
        self.name = "_" + name

    # 4 marks (1 for each line)
    def __set__(self, inst, value):
        if not isinstance(value, self.type):
            raise AttributeError("wrong type")
        setattr(inst, self.name, value)

    # 2 marks (1 for each line)
    def __get__(self, inst, cls):
        return getattr(inst, self.name)



# 5 marks (1 for each line)
# -1 mark if superclass of Meta is not type
class Meta(type):
    def __init__(cls, name, bases, attrs):
        for col, val in attrs.items():
            if isinstance(val, Field):
                val.setname(col)
```

c) You're helping a classmate with their assignment and the issue is that their user-defined class only supports exactly one instance. The symptom looks like this:

```
joe = User(name="Joe", age=12, height=5.4)
fred = User(name="Fred", age=23, height=6.2)
print(joe)
```

```
Fred: Age 23, Height 6.200000
```

Write down one line of code in `Field.__set__` that would produce this output (ignore the type checking part of `__set__`). [2 marks]

```
self.value = value
```

d) You're helping another classmate with their assignment and the issue is that all the fields have the same value, but multiple instances seems to be supported. The symptom looks like this:

```
Joe: Age Joe, Height Joe
Fred: Age Fred, Height Fred
```

Write down one line of code in `Field.__set__` that would produce this output. [2 marks]

```
inst.value = value
```

e) You're helping your friends with the same assignment again and the issue looks like this:

```
In Base.__init__:
RecursionError: maximum recursion depth exceeded
```

Describe (do not write code) what the problem may be and why it happened. [2 marks]

In setname, the same name is used for the value and the descriptor, causing Field.__set__ to be called infinitely. (must mangle the name to fix it, e.g. add underscore to name).

Question 7. **Rust Programming** [26 marks]

a) Given the following definition for Shape:

```
enum Shape {
    Rectangle(f64, f64), /* width, height */
    Triangle(f64, f64),  /* height, base */
    Circle(f64),         /* radius */
}
```

Complete the following function, `total_area`, which sums the area of a list of Shapes. Your solution must use the `fold` iterator adaptor and a `match` expression, and must not use any other control flow statements (e.g. if/else or loops). Contravention of these requirements is allowed, but will result in 50% loss of marks for this question. Assume `PI` is defined. [7 marks]

```
fn total_area(list: &Vec<Shape>) -> f64 {
    list.iter().fold(0.0, |ac, s| {     // 4 marks, 1 for each part
        ac + match s {                  // 1 mark
            Rectangle(w, h) => w * h,   // 2 marks for body of match
            Triangle(b, h) => b * h / 2.,
            Circle(r) => PI * r * r,
        }
    })

}
```

b) Given the following generic function, `sumsq`, which calculates the sum of squares:

```
fn sumsq<'a, T>(list: &'a Vec<T>) -> T {
    let mut total = Default::default();
    for val in list { total = total + val * val; }
    total
}
```

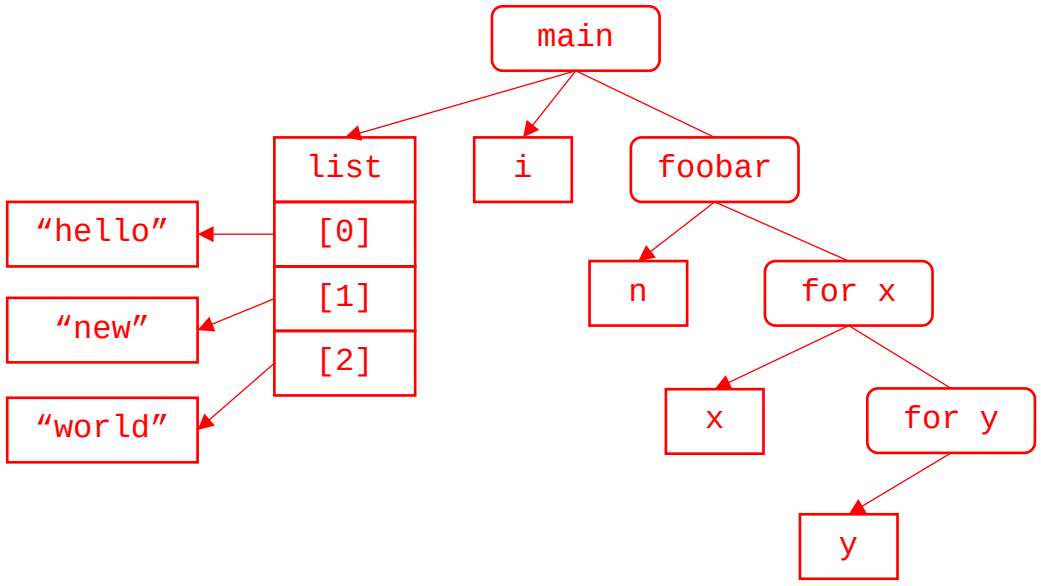
It currently does not compile because it's missing trait bounds. Circle the minimally required trait bound for this code to compile. [4 marks]

Copy	Clone	**Default**	Iterator
Add	AddAssign	**Mul**	PartialOrd

c) Given the following program:

```
struct Foo { name: String }
impl Foo { fn new(v: &str) -> Foo { Foo { name: v.to_string() } } }

fn foobar(n: i32, list: &Vec<Foo>) {
    for x in {0..n} {
        for (y, f) in list.iter().enumerate() {
            println!("{}.{}: {}", x, y, f.name); /* here */
        }
    }
}

fn main() {
    let i = 5;
    let list = vec![ Foo::new("hello"), Foo::new("new"),
                     Foo::new("world") ];
    foobar(i, &list);
}
```

Draw an ownership diagram in the form of a tree at the point when "0.2: world" is printed (the line with the comment "here"). Do **not** show borrowed objects. [8 marks]



[4 marks] 1 mark for i, n, x, y
[2 marks] for list and its elements
[2 mark] for x and y having the correct parents
-1 mark for incorrect parentage of i, n, and list

d) A Rustacean decided to write a program that will increment the values in a vector in parallel. It noticed that the code compiled, but the output is not correct and changes each time the program is run. Fix the code in the extra space provided so that it produces correct and consistent output. Assume the correct imports have been made. Note that not all blanks need to be filled. [6 marks]

```
fn main() {
    let v = vec![1, 2, 3];
    let data = Arc::new(Mutex::new(v));
    let mut handles = vec![];     // 2 marks
    for i in 0..99 {
        let data = data.clone();
        let handle = thread::spawn(move || {
            let mut data = data.lock().unwrap();
            data[i%3] += 1;
        });
        handles.push(handle);    // 2 marks
    }
    for h in handles {            // 1 mark
       h.join().unwrap();         // 1 mark
    }
    println!("{:?}", data);
}
```

e) What is the output of the program above? (just show the array elements) [1 mark]

[34, 35, 36]

Question 8. **Functional Programming** [12 marks]

a) The `lower` function takes a string slice, converts it to lower case, removes all non-alphabetic characters, and returns a String object. Complete this function in Rust using only iterators and iterator adaptors without using any control flow statements (e.g. if, else, for, .etc). Hint: `char::to_lowercase()` and `char::is_lowercase()` may be useful. [6 marks]

```
fn lower(word: &str) -> String {

  word.chars().map(|c| c.to_ascii_lowercase())
              .filter(|c| c.is_lowercase())
              .collect()

  1 mark for chars()
  1 mark for map
  1 mark for first closure
  1 mark for filter
  1 mark for second closure
  1 mark for collect

}
```

b) The `lower` generator takes in the name of a file, opens the file, splits the content into a list of words (by whitespace), sanitizes each word similar to what is done in part a., and yields each word. Complete this generator in Python. Your solution must make use of the `filter` function and a lambda function. Hint: the string method `islower()` and `lower()` may be helpful. [6 marks]

```
def lower(filename):
    with open(filename) as f:

       for word in f.read().split():
           yield ''.join(filter(lambda c: c.islower(),
                                word.lower()))


  1 mark for read().split()
  1 mark for ‘’.join
  1 mark for lambda
  1 mark for word.lower()
  1 mark for yield
  1 mark for filter

# example use
for w in lower("lorem.txt"):
    print(w)
```

Question 9. **Bonus Marks** [4 marks]

Each constructive answer is worth 1 mark.

a) Which topic in the lectures do you like **the most** in this year's offering of ECE326? Why?

b) Which topic in the lectures do you like **the least** in this year's offering of ECE326? Why?

c) Which lab assignment do you like **the most** in this year's offering of ECE326? Why?

d) Which lab assignment do you like **the least** in this year's offering of ECE326? Why?

[*Use the space below for rough work*]

END OF EXAMINATION