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 9 questions on 22 pages (including this page). The value of each part of each question is indicated. The total value of all questions is 170 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: _____ (16)

Q4: _____ (14)

Q5: _____ (15)

Q6: _____ (17)

Q7: _____ (32)

Q8: _____ (27)

Q9: _____ (15)

Total: _____ (170)

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. The `as` operator in Python and Rust does the same thing. **T** **F**

2. Python lambda function does not support multiple statements. **T** **F**

3. `set` and `dict` are both examples of unordered containers. **T** **F**

4. Depth-first-search, left to right, can fail if the method resolution order is ambiguous. **T** **F**

5. Protocol, known as interface in Java, does not promote code reuse. **T** **F**

6. Copy elision can cause C++ programs to behave differently depending on optimization level. **T** **F**

7. You can use the `cstdarg` library to extract arguments from a function with the signature `void foo(...);` **T** **F**

8. You can protect your C macros from side effects by adding parentheses around arguments that may have side effects, e.g. `#define foo(x) (x) * 3` **T** **F**

9. Rust has language support for concurrent programming. **T** **F**

10. Rust enum is an example of algebraic type. **T** **F**

11. To implement an iterator in Python, you must define `__iter__` and `__next__` in the same class. **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 or property, circle whether C++11, Python, or Rust supports it (more than one choice is possible). [7 marks]

i. Type-safe

C++11 Python Rust

ii. Iterator

C++11 Python Rust

iii. Contravariant Parameter

C++11 Python Rust

iv. Block Scope

C++11 Python Rust

v. Enumerated Types

C++11 Python Rust

b) Which of the following keyword can be used as part of an expression in Rust? [4 marks]

i. `if`

ii. `let`

iii. `loop`

iv. `match`

v. `use`

c) Which of the following functions are examples of a `fold` function? [4 marks]

i. `reverse`

ii. `sum`

iii. `any`

iv. `map`

v. `zip`

d) Which of the following statements are true regarding ownership and borrowing? [4 marks]

i. You cannot access a variable after it has been moved.

ii. You cannot change a variable while there are immutable references to it.

iii. You can mutably borrow multiple distinct slices of a sequence at one time.

iv. You can have multiple immutable references to a variable at one time.

v. You can change an immutable variable into a mutable one after moving it.

e) Which of the following describes the C preprocessor macros? [4 marks]

i. Domain-specific

ii. Turing-complete

iii. Reflective

iv. Functional

v. Manifest typing

Question 3. **Short Answers** [16 marks]

a) Spin lock is a type of lock that causes a thread to try to acquire the lock in a tight loop. For example:

```
while(test_and_set(&lock) != 0);
```

`test_and_set` is an atomic operation that sets the value of lock to 1 and simultaneously returns the old value.

i. Would you use a spinlock on a uniprocessor system? If yes, is there a precondition for its use, and what is it? If no, why not? [2 marks]

ii. Would you use a spinlock on a multiprocessor system? If yes, is there a precondition for its use, and what is it? If no, why not? [3 marks]

b) In C++, suppose Fred is a subclass of Flob, and foo() is a virtual method in class Bar that returns a pointer to class Flob. Is it legal for a subclass of Bar to override foo() and change the return type to a pointer to class Fred? What is the name of this language feature, regardless of whether or not C++ supports it? [2 marks]

c) Show an example of shadowing for C++, Python, and Rust. For example:

```
class A: foo = 0
class B(A): foo = 1
```

You should circle the variable that is shadowed. You may not use the example above, or a derivative of it, to receive any marks for Python. Your answer must clearly show the scope of each variable involved. Add comments if it helps clarify your example. [9 marks]

C++

Python

Rust

Question 4. Object-Oriented Programming [14 marks]

You are working with a graphical toolkit that provides the following interface and widgets:

```
struct Input {
    virtual const string & getvalue() const=0;
    virtual void draw() const=0;
};

class Select : public Input {
    vector<string> choices;
    int index;
public:
    void set_index(int i) { index = i; }
    virtual const string & getvalue() const override;
    virtual void draw() const override;
};

struct Text : public Input {
    string value;
public:
    virtual const string & getvalue() const override;
    virtual void draw() const override;
};
```

You wish to implement a new class, `Suggest`, which allows the users to narrow down on the choices available in `Select` as they type (think Google Suggest).

a) Draw the data layout of the `Suggest` class using multiple inheritance, inherit `Select` first, then `Text`. Note: `sizeof(vector<string>)` is 24, and `sizeof(string)` is 32. [5 marks]

b) What is the size of the `Suggest` class implemented using multiple inheritance? [1 mark]

c) Define the `Suggest` class using multiple inheritance. Next implement the interface functions `getvalue` and `draw`. `getvalue` should return the value returned by `Select`, and `draw` should draw the `Select` widget first before `Text`. [4 marks]

d) Define the `Suggest` class again using composition. Note that `Suggest` must still be a subclass of `Input`. Do not implement either interface functions (just declare them). [4 marks]

Question 5. **Template Metaprogramming** [15 marks]

a) Complete the template metafunction, `is_mapper`, that checks whether type F is a function or a functor that takes one parameter: type T, and returns type T. [9 marks]

```
is_same<int, decltype(123)>::value // this evaluates to true
```

Hint: you should use make use of `is_same`, shown above, to check the return type of type F.

```
template<typename T, typename F, typename=void>
struct is_mapper : false_type {};
```

b) Write a template **functor**, `AddN`, which adds *n*, of type T, to the argument of type T passed to the functor. The value *n* is initialized through `AddN`'s constructor. [6 marks]

```
is_mapper<AddN<int>, int>::value; // this needs to be true
```

Question 6. **Reflective Programming** [17 marks]

a) Python property can actually be implemented using a descriptor. Implement a simplified version of property, named Property, that supports calling a read function `fget` upon attribute read, calling a write function `fset` upon attribute write, but does not support delete. If `fget` is `None`, then the attribute is write-only. If `fset` is `None`, then the attribute is read-only. In either of these cases, raise `AttributeError` with the message, "operation not supported". [10 marks]

```
class Property:
    def __init__(self, fget=None, fset=None):
```

b) Can your Property class be used as a decorator? If yes, show an example of how you would use it. If no, explain the error that would manifest. [3 marks]

c) Describe two things that can only be done with metaclasses and not by any other language features in Python. [4 marks]

Question 7. **Rust Programming** [32 marks]

a) If you try to compile the code below, it will generate errors/warnings. Identify the line(s) where error(s) would occur by specifying the line number(s), and if possible, write the code that would fix it. If a print statement would cause the error, write "DELETE" in the table. [10 marks]

```
1    fn create_model(company: String) -> String {
2        let model = String::from("Default, ");
3        if company == "Boeing" {
4            model = String::from("747, ");
5        }
6        else if company == "Airbus" {
7            model = String::from("A330, ");
8        }
9        model.push_str(&company);
10       model
11   }
12
13   fn create_airbus() -> &str {
14       let some_string = "Airbus";
15       &some_string
16   }
17
18   fn create_plane() {
19       let plane1 = create_airbus();
20       let plane2 = String::from("Boeing");
21       let plane3 = create_model(plane1);
22       let plane4 = create_model(plane2);
23       println!("Plane 1: {}", plane1);
24       println!("Plane 2: {}", plane2);
25       println!("Plane 3: {}", plane3);
26       println!("Plane 4: {}", plane4);
27   }
28
29   fn main() {
30       let mut height = 100;
31       let my_plane = "777, Boeing";
32       {
33           let b = my_plane;
34       }
35       println!("Plane is a: {}", b);
36       println!("Currently at {}", height);
37       {
38           let c = 2;
39           height += c;
40       }
41       println!("Ascending to {}", height);
42       create_plane();
43   }
```

Line #	Corrected line or "DELETE"

b) Assuming all the errors were fixed, write the output of the program. [2 marks]

c) Consider the following code:

```
 trait Talk {
     fn noise(&self) -> &str {
         "woof"
     }
 }

trait Animal {
    fn animal_type(&self) -> &str;
}

struct Dog { name: String }
struct Cat;

fn main() {
    let dog = Dog::new("Fido");
    let cat = Cat;
    println!("The dog went {}", dog.noise());
    println!("{} is a {}", dog.name, dog.animal_type());
    println!("The cat went {}", cat.noise());
}
```

Write the implementations for Dog and Cat so that the program output looks like this. [9 marks]

```
The dog went woof
  Fido is a dog
The cat went meow
```

d) Given the following generic function, it currently does not compile because it is missing some trait bounds. Circle the minimal required trait bounds that will enable it to compile, [4 marks]

```
fn animal_party<T>(host: &T, guests: Vec<T>) {
    let mut i = 0;
    for guest in guests {
        if *host < guest {
            println!("{} {}", host.noise(), guest); /* part e */
        }
        else { drop(guest); i += 1; }
    }
    println!("{} bounced", i);
}
```

Copy	Drop	Display	Debug
PartialOrd	PartialEq	Animal	Talk

e) Suppose we add the following code to the bottom of the previously defined main function:

```
fn main() {
    /* code from part c */
    let dogs = vec![ Dog::new("Aaron"), Dog::new("Joey"),
                     Dog::new("Bailey") ];
    animal_party(&dog, dogs);
}
```

Draw an ownership diagram in the form of a tree at the point when “woof Joey” is printed (the line with the comment “part e”). Do not draw references to borrowed objects. [7 marks]

Question 8. Concurrent Programming [27 marks]

In a peculiar home with mice and cats, they share an extra-large bowl of food together. The rules at this house is that the cats must eat alone, while the mice can eat together, up to 6 of them at once. When a cat wants to eat, it has priority over any mouse that wants to eat. The mice who are already eating must finish up and leave. If another cat is eating, the other cat must wait for its turn.

Given the following shared data structure:

```
enum Eater { NONE, CATS, MICE, }
struct Bowl {
    eater: Eater, // which kind of animal is eating right now
    count: i32,   // number of animals eating together (mice only)
    ncats: i32,   // number of cats waiting
}
```

a) Given the following code snippet:

```
1  struct Monitor {
2      mutex: Mutex,
3      cv: Condvar,   // waiting for the bowl
4  }
5
6  const NCATS = 2;
7
8  fn main() {
9      let mut threads = vec![];
10     let bowl = Bowl { eater: NONE, count: 0, ncats: 0 };
11     let monitor = Rc::new(Monitor {
12         mutex: Mutex::new(bowl),
13         cv: Condvar::new(),
14     });
15
16     for i in 1..=NCATS {
17         threads.push(thread::spawn(|| {
18             for _ in 0..10 {
19                 cat_eat(i, &monitor);
20             }
21         }));
22     }
23
24     /* create mouse threads here */
25
26     for child in threads {
27          child.join().unwrap();
28     }
29 }
```

The above code currently does not compile. Identify the line(s) where error(s) would occur by specifying the line number(s), and if possible, write the code that would fix it. [10 marks]

Line #	Corrected Line

b) Complete the `cat_eat` function by filling in the blanks. [5 marks]

```
fn cat_eat(i: u64, monitor: ____________________________________) {
    let Monitor {mutex, cv} = &**monitor;
    let mut bowl = mutex.lock().unwrap();

    while ____________________________________________________ {
        bowl.ncats += 1;

        bowl = ____________________________________; // wait its turn
        bowl.ncats -= 1;
    }
    bowl.eater = CATS;
    bowl.count = 1;

    ____________________________________________; // unlock
    cat_eating();   // lock should not be held during this call
    let mut bowl = mutex.lock().unwrap();
    bowl.eater = NONE;
    bowl.count = 0;

    ____________________________________________; // wake up everyone
}
```

c) Complete the `mouse_eat` function in the spaces provided below. [12 marks]

```
fn mouse_eat(/* function parameters redacted */)
{
    let Monitor {mutex, cv} = &**monitor;
    let mut bowl = mutex.lock().unwrap();









    mouse_eating(); // lock should not be held during this call
    let mut bowl = mutex.lock().unwrap();









}
```

Question 9. **Functional Programming** [15 marks]

a) Complete the following generic function, `average`, such that it returns the arithmetic average of the type parameter T. Your solution must make use of the `fold` iterator adaptor, and must be able to handle an empty list correctly without crashing. [7 marks]

```
fn average<T>(list: & Vec<T>) -> T
where T: Copy + Add<Output=T> + Default + From<u32> + Div<Output=T>
{




}
```

b) Given a CSV (comma-separated values) file where the first row contains the header of each column and the remaining rows are values. For example:

```
last, first, player_nr
Lowry, Kyle, 7
Siakam, Pascal, 43
Gasol, Marc, 33
```

Write a generator that takes in the name of a file, opens the file, get the header row, and for each subsequent row, yields a dictionary where the keys are the header values. For example, the first dictionary that should be returned is:

```
{'last': 'Lowry', 'first': 'Kyle', 'player_nr': '7'}
```

You are required to make use of the `zip` built-in function. You may assume the file is correctly formatted, but you must *strip* whitespaces for each input token. (e.g. from ‘ last’ to ‘last’) [8 marks]

```
def csv_to_dict(filename):
```

[*Use the space below for rough work*]

END OF EXAMINATION