Programming Languages **Practice Final Solution** ECE326, Fall 2020

University of Toronto

Duration: 2 hours 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 15 pages (including this page). The value of each part of each question is indicated. The total value of all questions is 95 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:	________	(6)
Q2:	________	(10)
Q3:	________	(6)
Q4:	________	(10)
Q5:	________	(17)
Q6:	________	(15)
Q7:	________	(13)
Q8:	________	(5)
Q9:	________	(8)
Total:	________	(90)

Question 1. In the Shadow [6 marks]

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

```
class A: $\boxed{foo}$ = 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.

C++

```
int $\boxed{x}$ = 5;
{
    int x = 6;
}
```

Python

```
$\boxed{x}$ = 5

def foo():
    x = 6
```

Rust

```
let $\boxed{x}$ = 5;

let x = x + 2;
```

Question 2. Concurrent Programming [10 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]

No – you should always go to sleep immediately if you cannot acquire block because you would otherwise hog CPU and not allowing other threads to run in the meantime.

This answer is also accepted:

Yes – if the wait time is expected to be less than the time to perform a context switch.

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]

Yes – as long as the wait time is less than time to perform a context switch or if there are few threads running (e.g., there are no threads to run even if you go to sleep).

b) 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. [4 marks]

```
fn main() {
    let v = vec![1, 2, 3];
    let data = Arc::new(Mutex::new(v));
     let mut handles = vec![];      // 1 mark
    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);     // 1 marks
    }
     for h in handles {             // 1 mark
        h.join().unwrap();          // 1 mark
     }
    println!("{:?}", data);
}
```

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

[34, 35, 36]

Question 3. Method Resolution Order [6 marks]

Given the following class hierarchy:

```
class A: pass                    class E(A,B,C): pass
class B: pass                    class F(B,C,D): pass
class C: pass                    class G(E,F): pass
class D: pass                    class H(G): pass
```

Compute the method resolution order for the class H using C3 linearization. You must show your steps for the classes G and H to receive full marks. For classes A to F, just show the results.

```
L(A)=(A,o)
L(B)=(B,o)
L(C)=(C,o)
L(D)=(D,o)
L(E)=(E,A,B,C,o)
L(F)=(F,B,C,D,o)
L(G)=(G,merge((E,A,B,C,o),(F,B,C,D,o),(E,F)))
L(G)=(G,E,merge((A,B,C,o),(F,B,C,D,o),(F)))
L(G)=(G,E,A,merge((B,C,o),(F,B,C,D,o),(F)))
L(G)=(G,E,A,F,merge((B,C,o),(B,C,D,o)))
L(G)=(G,E,A,F,B,merge((C,o),(C,D,o)))
L(G)=(G,E,A,F,B,C,merge((o),(D,o)))
L(G)=(G,E,A,F,B,C,D,merge((o),(o)))
L(G)=(G,E,A,F,B,C,D,o)
L(H)=(H,G,E,A,F,B,C,D,o)
```

Question 4. Reflective Programming [10 marks]

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”.

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

        self.fget = fget
        self.fset = fset


    def __get__(self, inst, cls):
        if self.fget is None:
            raise AttributeError("operation not supported")
        return self.fget(inst)


    def __set__(self, inst, value):
        if self.fset is None:
            raise AttributeError("operation not supported")
        self.fset(inst, value)
```

Question 5. Rust Programming [18 marks]

a) 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. [7 marks]

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

```
impl Animal for Dog {
    fn animal_type(&self) -> &str {
        "dog"
    }
}

impl Talk for Dog {}

impl Dog {
    fn new(name: &str) -> Dog {
        Dog { name: name.to_string() }
    }
}

impl Talk for Cat {
    fn noise(&self) -> &str {
        "meow"
    }
}
```

b) 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]

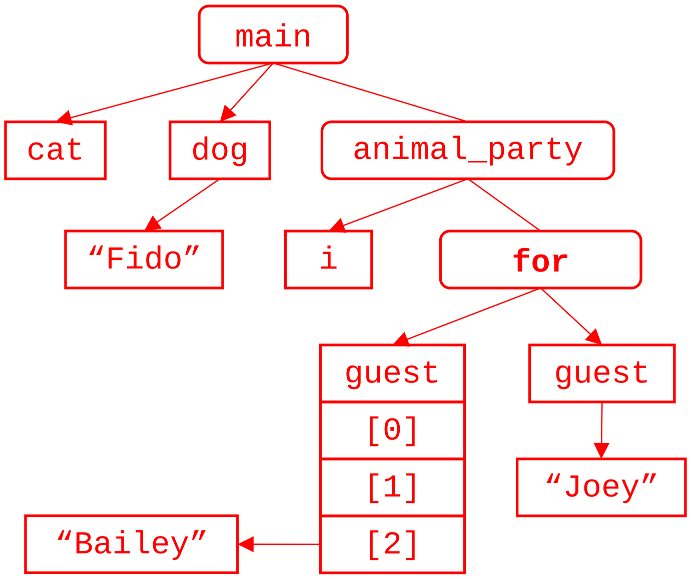
```
fn animal_party<T>(host: &T, guests: Vec<T>) {
    let mut i = 0;
    for guest in guests {
        if *host < guest {
            println!("{} {}", host.noise(), guest); /* part e */
        }
        else { drop(guest); i += 1; }
    }
    println!("{} bounced", i);
}
```

Copy	Drop	[✓ Display]	Debug
[✓ PartialOrd]	PartialEq	Animal	[✓ Talk]

c) 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 6. Template Programming [15 marks]

Implement a template class for a binary search tree that has two member functions: *insert* and *print*.

- `insert` will attempt to add a new element to the binary search tree. It shall return false if duplicate is found, and true if insertion is successful.
- `print` will print all elements of the tree using in-order traversal, space delimited.

Your solution also requires a default constructor and a destructor that does not leak memory.

```
template<typename T>
class BSTree
{
    struct Node
    {
        T value;
        Node * left;
        Node * right;

        Node(const T & v) : value(v), left(nullptr),
            right(nullptr) {}
        ~Node() { delete left; delete right; }
    } * root;

protected:
    /* defined on the next page */
    bool insert_recursive(Node *& curr, const T & v);
    void print_recursive(Node * curr) const;

public:
    BSTree() : root(nullptr) {}
    ~BSTree() { delete root; }

    /* defined outside of class definition */
    bool insert(const T & v);
    void print() const;
};
```

```
template<typename T>
bool BSTree<T>::insert(const T & v) {
    return insert_recursive(root, v);
}

template<typename T>
bool BSTree<T>::insert_recursive(Node *& curr, const T & v)
{
    if (curr == nullptr) {
        curr = new Node(v);
        return true;
    }
    else if (v < curr->value) {
        return insert_recursive(curr->left, v);
    }
    else if (v > curr->value) {
        return insert_recursive(curr->right, v);
    }

    return false;
}



template<typename T>
void BSTree<T>::print() const {
    print_recursive(root);
    std::cout << std::endl;
}

template<typename T>
void BSTree<T>::print_recursive(Node * curr) const
{
    if (curr != nullptr) {
        print_recursive(curr->left);
        std::cout << curr->value << " ";
        print_recursive(curr->right);
    }
}
```

Question 7. Seven Mathematicians [13 marks]

a) Write a variadic template function, `is_factor`, that returns true whenever the first argument is a factor of any of the subsequent arguments. For example:

```
int n = 120;
is_factor(n, 9, 13, 7); // returns false
is_factor(n, 60, 17);   // returns true
```

Your solution must support any integral type that implements the modulo operator. [6 marks]

```
template<typename T>
bool is_factor(T n) {
    return false;
}

template<typename T, typename ... Args>
bool is_factor(T n, T f, Args ... args) {
    if (n % f == 0)
        return true;
    return is_factor(n, args...);
}
```

b) The Seven Dwarfs are secretly mathematicians who have affinity towards large numbers that are factors of any of their favorite numbers. This information is kept in a macro constant as shown here:

```
#define DWARFS \
    D(Doc, 11) \
    D(Happy, 7, 9) \
    D(Sneezy, 5, 6, 7) \
    D(Sleepy, 4, 17) \
    D(Bashful, 3, 8) \
    D(Grumpy, 13) \
    D(Dopey, 10)
```

Use the X-macro technique and the macro above to populate an enum named Dwarf. [2 marks]

```
enum Dwarf {
#define D(name, ...) name,
    DWARFS
#undef D
};
```

c) Use the X-macro technique and the DWARFS macro to complete the following function, that takes a number and returns a vector of Dwarfs that like this number. Hint: use `is_factor`. [5 marks]

```
std::vector<Dwarf> who_likes_this_number(int num)
{
    std::vector<Dwarf> ans;

#define D(name, ...) if (is_factor(num, ##__VA_ARGS__)) { \
        ans.push_back(name); \
    }
    DWARFS
#undef D

    return ans;
}
```

Question 8. Lazy Evaluation [5 marks]

Given the following Python function:

```
def oscillate(n):
    a = list(range(0, -n*2, -1))
    b = [1, -1] * n
    c = map(lambda t: math.sqrt(t[0]*t[1]), zip(a, b))
    return sum(list(c)[1::2])
```

a) There would be difference between the output of the above function if Python were to use lazy evaluation instead of eager evaluation? Show the difference and explain why there is a difference. [2 marks]

Normally, with eager evaluation, the function would crash on ValueError because the sqrt function is applied to all elements of the zip(a, b), half of which are negative numbers and the sqrt of negative numbers is not permissible. However, with lazy evaluation, since the slice only collects the valid values, the program would not crash because it does not evaluate expressions that is not used. Hence, it would theoretically return the result of the following mathematical expression:

$$\sum_{i=0}^{n-1} \sqrt{2i+1}$$

b) Write an alternative implementation of the function that would result in the same program behaviour as if Python were using lazy evaluation. [3 marks]

```
def oscillate(n):
    a = range(0, -n*2, -1)
    b = [1, -1] * n
    z = list(zip(a, b))[1::2]
    c = map(lambda t: math.sqrt(t[0]*t[1]), z)
    return sum(c)
```

Question 9. Python Functional Programming [8 marks]

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 function that takes in the name of a file, opens the file, get the header row, and for each subsequent row, create a dictionary where the keys are the header values, i.e. a list of dictionaries. 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 `map` and `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’)

```
def csv_to_dict_list(filename):




    output = list()
    with open(filename) as f:
        headers = list(map(lambda x: x.strip(), next(f).split(",")))
        for line in f:
            values = list(map(lambda x: x.strip(), line.split(",")))
            output.append(dict(zip(headers, values)))
    return output
```

[*Use the space below for rough work*]

END OF EXAMINATION