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: Ruler and examiner approved aid sheet are allowed.

MARKING GUIDE

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

Q1: _____ (10)

You may remove the aid sheet from the back of this test book. Do not remove any other 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.

Q2: _____ (10)

Q3: _____ (7)

This exam consists of 8 question on 15 pages (including this page). The value of each part of each question is indicated. The total value of all questions is 75 marks.

Q4: _____ (8)

Q5: _____ (9)

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!

Q6: _____ (9)

Q7: _____ (12)

Work independently.

Q8: _____ (10)

Total: _____ (75)

Question 1. What's the Point? [10 marks]

Given the following programming language terms:

1. Abstraction
2. Code Reuse
3. Performance Optimization

4. Memory Safety
5. Type Safety
6. Thread Safety

For each one of the programming language paradigms or features, first state its primary purpose (choose from 1 to 6), then explain your choice. Note: you may only choose 1 option out of the 6.

a) Functions (example answer)

   2 (code reuse) – because a function can be called in multiple parts of the source code to perform same or similar tasks, allowing the functionality to be reused without duplicating code.

b) Concurrent Programming

c) Macro Programming

d) Atomic Variables

e) Tagged Union

f) Constexpr functions

Question 2. **Pure or Not?** [10 marks]

For the following C++ functions, state whether they are pure functions. If no, explain why not.

```
int x = 5;                              // global variables
constexpr int y = 2;
```

`int f(int a) {`
`    int x = a + 3;`
`    return x;`
`}`	

`int g(int & a, int & b) {`
`    a += 2;`
`    return a + b;`
`}`	

`template<int N>`
`int h(const int (&a)[N]) {`
`    int total = 0;`
`    for (int i = 0; i < N; i++)`
`        total += a[i];`
`    return total;`
`}`	

`int k(int a) {`
`    int b = a++;`
`    return b * a;`
`}`	

`int p(const int * a, int i) {`
`    int n = a[0];`
`    return a[i] + n;`
`}`	

`int q(int a) {`
`    // returns random number`
`    int b = rand();`
`    return b % a;`
`}`	

`int t() {`
`    return y;`
`}`	

Question 3. **Palindrome** [7 marks]

In C++11, write a constexpr function named `palindrome` to check whether a string is a palindrome. A palindrome is a word or a phrase that reads the same forward and backward, e.g., "madam", "top spot". Your solution must be able to ignore space characters, and your code must compile using the `--std=c++11` flag. You may assume all letters in the string are lower case, and you do not need to deal with punctuations. Hint: you may need a helper function.

Question 4. Python 2.2 [8 marks]

Given the following class definitions:

```
class A:
    def __init__(self, a, **kwargs):
        self.x = a
        super().__init__(**kwargs)
        self.z = a

class B:
    def __init__(self, b, **kwargs):
        self.y = b
        super().__init__(**kwargs)

class D(B):
    def __init__(self, d, **kwargs):
        self.x = d
        super().__init__(**kwargs)
        self.y = d

class M(A, B):
    def __init__(self, m, **kwargs):
        self.x = m
        super().__init__(**kwargs)
        self.z = m

class N(D, A):
    def __init__(self, n, **kwargs):
        super().__init__(**kwargs)
        self.z = n

class Q(M, N):
    def __init__(self):
        self.y = 1
        super().__init__(a=2, b=3, d=4, m=5, n=6)

foo = Q()
```

Assume this code is run under Python 2.2, where the method resolution order is calculated using depth-first search, left to right, *with refinement* (i.e., remove earlier duplicates).

a) What is the linearization of the class Q? For class A, B, D, M, N, you may show results only. For class Q, you must show work to receive full marks. [5 marks]

b) What are the values of the following attributes? [3 marks]

foo.x	foo.y	foo.z

Question 5. **Rust Tournament** [10 marks]

You are simulating a knockout tournament in a Rust program. Given the following struct definition for a Team:

```rust
struct Team {
    name: String,
    power: i32,
}
```

a) Write a constructor for Team in the form of a static method named `new` that takes two parameters: `name`, which is a *string literal* (not a String object), and `power`, which is an i32 integer. [2 marks]

b) Complete the function, `match_up`, which takes two teams, `home` and `away`, by reference, and returns the team with higher power. On a tie, return the home team. [2 marks]

c) Complete the function, `round`, which takes two vectors of teams by references (i.e. vectors of references), do an element-wise match-up of the two set of teams using the `match_up` function, and return a vector of the winning teams *by reference*. You must complete this function in one line of Rust code, without using a semicolon within the function body. [5 marks]

Question 6. **Problem with Homonyms** [9 marks]

Given the following two classes in C++.

```
class Prisoner {                          class Method {
  Person * person;                          int i = 0;
public:                                   public:
  /* … other functions …*/                  /* … other functions …*/
  void execute() {                          void execute() {
    person->kill();                           i++;
    delete person;                            std::cout << "i = " << i
    person = nullptr;                                        << std::endl;
  }                                         }
};                                        };
```

And a template function which will repeatedly call the execute member function.

```
template<typename T>
void repeat(T & repeatable, int n) {
    for (int i = 0; i < n; i++)
        repeatable.execute();
}
```

Below is the output when running the repeat function on an object of the above two classes:

```
[main.cpp for Prisoner]
Person * person = new Person();
Prisoner prisoner(person);
repeat(prisoner, 3);


Program Output:
He's Dead, Jim
Segmentation fault (core dumped)



[main.cpp for Method]
Method method;
repeat(method, 3);


Program Output:
i = 1
i = 2
i = 3
```

a) Convert the C++ code to Rust while maintaining the same program structure. If there is a language feature that can avoid the problem shown above, use it to fix the problem. If not, show that the same problem will manifest. Briefly explain your answer. [5 marks]

b) Convert the C++ code to Python while maintaining the same program structure. If there is a language feature that can avoid the problem shown above, use it to fix the problem. If not, show that the same problem will manifest. Briefly explain your answer. [4 marks]

Question 7. Roman Numerals [12 marks]

The following macro constant contains a list of macro invocations that maps roman numeral symbols to their values.

```
/* N(SYMBOL, VALUE, SUFFIX) */
#define ROMAN_NUMERALS \
    N(I,    1, S(V) S(X)) \
    N(V,    5) \
    N(X,   10, S(L) S(C)) \
    N(L,   50) \
    N(C,  100, S(D) S(M)) \
    N(D,  500) \
    N(M, 1000)
```

For example, the symbol X maps to the value 10, and the symbol M maps to 1000. The roman numeral "III" is the integer value 3, and "MLXXI" equals to $1000 + 50 + 10 + 10 + 1 = 1071$.

Roman numerals use a subtractive notation where the value of two consecutive symbols is the difference between their values. For example, IV is $5 - 1 = 4$ (i.e., $V - I$), and XC is $100 - 10 = 90$ (i.e., $C - X$). When subtractive notation is allowed for a particular symbol, the N macro function takes an optional third parameter which contains an inner list of macro invocations for each suffix symbol. For example, when C is followed by D or M, subtractive notation is applied, but not for L.

Complete the function, `roman_to_integer`, which will convert roman numerals, encoded as null-terminated strings, to integers. You must use the X-macro technique to receive full marks. Note: while processing the string, if it contains any non-roman numeral characters, the function should return 0 immediately. You may otherwise assume that all input will be roman numerals in standard form.

```
/* Hint: create a mapping between the symbols and the values here */




```

```
unsigned roman_to_integer(const char * str)
{
    const unsigned len = strlen(str);
    unsigned idx = 0;
    unsigned val = 0;

    while (idx < len)
    {




























    }

    return val;
}
```

Question 8. **Database Row** [10 marks]

You are writing a template class to represent a database row object. Each row has many columns, and each column has a type. Columns inside a row may have different types.

```
template <typename ... ColumnTypes> struct Row;
```

a)   Write the recursive template definition for struct Row. Remember to define a constructor. [4 marks]

b)   The struct `ColumnOf` can be used to get the column type given a row type. For example:
     `ColumnOf<Row<int, float>, 1>::Type` would be the type float. Read the following code
     and add two statements to `struct Row` definition above to make it work with `ColumnOf`. Note:
     inside a template definition, when referring to a type defined inside another template, you must
     prefix the type (known as dependent name) with the keyword `typename`. [2 marks]

```
template <typename T, int N>
struct ColumnOf {
    using Type = typename ColumnOf<typename T::ParentType,
                                   N - 1>::Type;
};

template <typename T>
struct ColumnOf<T, 0> {
    using Type = typename T::ValueType;
};
```

c) Write a template class named Column with a *static template member function* named Get that will allow the programmer to retrieve a particular column from a row object. For example:

```
using BankRow = Row<const char *, const char *, int, double>;
auto row = BankRow("Mike", "Qin", 31, 1000.25);
int age = Column<2>::Get(row);   // returns 31
```

Hint: `ColumnOf` will be very helpful when extracting values from the row object. [4 marks]

```
template<int N> struct Column
```

[*Use the space below for rough work*]

END OF EXAMINATION