School of Science & Technology

Department of Computer Science

BSc/MSci Computer Science
BSc/MSci Computer Science with Computer Games Technology
MSci Computer Science with Cyber Security
MSci Data Science

IN2029: Programming in C++

Stage 2 Examination

January 2023

Examination duration: **90 minutes**

Division of marks: Full marks may be obtained for correct answers to **Two** of the **Three** questions. If more than **Two** questions are answered the best **Two** will be counted.

Other instructions:

BEGIN EACH QUESTION ON A FRESH PAGE

Number of answer books to be provided: 1
Calculators permitted: Casio FX-83/85 MS/ES/GT+/GTX ONLY
Examination duration: 90 minutes
Dictionaries permitted: None
Can question paper be removed from the examination room: No
Additional materials: None

External Examiner:	Prof. Keshav Dahal
Internal Examiner:	Dr Ross Paterson

1. (a) Consider the following function definition:

```
int f(int i, int & j) {
    i = i+j;
    j = 2*i;
    return i+j;
}
```

What is printed by the following code:

```
    int x = 3, y = 5;
    cout << f(x, y) << '\n';
    cout << "x = " << x << ", y = " << y << '\n';
```

[10 Marks]

(b) Write a function that takes a list of `doubles` and prints them out, one per line. **[10 Marks]**

(c) Using the following library algorithm:

count_if(b, e, p) returns the number of elements in the range $[b, e)$ for which p is `true`.

Write a function that takes a list of strings and returns the number of strings in the list that have a length of at least 6. **[20 Marks]**

(d) Using the following library algorithm:

find(b, e, v) searches the range $[b, e)$ and returns an iterator pointing at the first occurrence of v, if there is one, or e if there is not.

(i) Write a function that takes a list of strings and a search string, and returns a `bool` indicating whether the search string occurs in the list. **[20 Marks]**

(ii) Write a function that takes a list of strings, a search string and a replacement string, and modifies the list by replacing the first element equal to the search string with the replacement string. If there is no such occurrence the list should be unchanged. **[20 Marks]**

(iii) Write a variant of the previous function that replaces all the elements equal to the search string with the replacement string. **[20 Marks]**

2. (a) Define a class `character` for non-player characters in a game, each with a name (a string), an initial health (a positive whole number) and a current health (also a positive whole number). Your class should include

    (i) a constructor taking the name of the character and an initial health.

    (ii) accessor functions for the name and current health.

    (iii) a member function to decrease the character's health by one, but not below zero.

    (iv) a member function to increase the character's health by one, but not above that character's initial health. However, if a character's health is already zero (meaning the character is dead), the health should not be increased.

    [35 Marks]

   (b) Define a `world` class containing an arbitrary number of `character` objects. Your class should include member functions to

    (i) add a new `character`.

    (ii) increase the health of all living characters by one.

    (iii) given a name, decrease the health of all characters with that name.

    [35 Marks]

   (c) Using the following library algorithm:

    count_if(b, e, p) returns the number of elements in the range $[b, e)$ for which p is `true`.

    write a member function of the `world` class that returns the number of live characters, that is, those with a positive health. **[30 Marks]**

3. Consider the following class:

```
class scorer {
public:
    // Combined score of a non-empty vector.
    virtual double score(const vector<double> &v)
        const = 0;
};
```

(a) Describe the effects of the two uses of `const` in this class. **[10 Marks]**

(b) Write a derived class of `scorer` in which the member function `score()` returns the average of the numbers in the vector. **[30 Marks]**

(c) Write a derived class of `scorer` in which the member function `score()` returns the largest of the numbers in the vector. **[30 Marks]**

(d) Write a function that takes two arguments:

- a `scorer` giving a way of generating scores, and
- a `map<string, vector<double>>` giving non-empty collections of scores associated with names,

and prints each name with the summary score calculated from the associated vector using the `scorer`. Take particular care with how the arguments are passed. **[30 Marks]**

End of paper

Marking Scheme

1. (a) required answer: 10 marks

```
24
x = 3, y = 16
```

Marking: return value [4], x [3], y [3].

Working (not required, but may gain partial marks if final answer wrong):

- i = 3
- i = 8
- j(y) = 16
- return 24

(b) 10 marks

```
void print_list(const list<double> &l) {
    for (auto x : l)
        cout << x << '\n';
}
```

Alternative iterator version:

```
void print_list(const list<double> &l) {
    for (auto p = l.cbegin(); p != l.cend(); ++p)
        cout << *p << '\n';
}
```

Marking: signature [5, including 2 for const reference], loop [3], print [2].

(c) We need an external function 20 marks

```
bool long_word(const string &s) {
    return s.size() >= 6;
}
```

Then our function is

```
int long_count(const list<string> &l) {
    return count_if(l.cbegin(), l.cend(), long_word);
}
```

Marking: `long_word` [10, including 2 for const and reference], `long_count`: signature [4, including 2 for const and reference], call to `count_if` [6]. Maximum 10 marks for an otherwise correct function that does not use `count_if()`.

(d) (i) 20 marks

```
bool search(const list<string> &l, const string &s) {
    return find(l.cbegin(), l.cend(), s) != l.cend();
}
```

Marking: signature [5], `find` [10], test [5]. Maximum 10 marks for an otherwise correct function that does not use `find()`.

(ii) 20 marks

```
void replace_first(list<string> &l,
        const string &s, const string &r) {
    auto p = find(l.begin(), l.end(), s);
    if (p != l.end())
        *p = r;
}
```

Marking: signature [5], `find` [8], test [4], update [3]. Maximum 10 marks for an otherwise correct function that does not use `find()`.

(iii) 20 marks

```
void replace_all(list<string> &l,
        const string &s, const string &r) {
    for (auto p = find(l.begin(), l.end(), s);
         p != l.end();
         p = find(++p, l.end(), s))
        *p = r;
}
```

An equivalent `while` loop is equally acceptable:

```
void replace_all(list<string> &l,
        const string &s, const string &r) {
    auto p = find(l.begin(), l.end(), s);
    while (p != l.end()) {
        *p = r;
        ++p;
        p = find(p, l.end(), s);
    }
}
```

Marking: signature [5], `find` [5], loop [10]. Maximum 10 marks for an otherwise correct function that does not use `find()`.

2. (a) 35 marks

```
class character {
    const string _name;
    const int init_health;
    int curr_health;

public:
    character(const string &n, int h) :
        _name(n), init_health(h), curr_health(h) {}

    const string & name() const { return _name; }
    int health() const { return curr_health; }

    void decrease() {
        if (curr_health > 0)
            --curr_health;
    }

    void increase() {
        if (curr_health > 0 && curr_health < init_health)
            ++curr_health;
    }
};
```

Marking: private [2] data members [3], constructor [10, including 2 for const ref], accessors [8], increase [6], decrease [6].

(b) 35 marks

```
class world {
    vector<character> chars;

public:
    void add_character(const character &c) {
        chars.push_back(c);
    }

    void increase() {
        for (auto &c : chars)
            c.increase();
    }
    void decrease(const string &n) {
        for (auto &c : chars)
            if (c.name() == n)
                c.decrease();
    }
};
```

Iterator versions of the last two member functions are equally acceptable:

```
    void increase() {
        for (auto p = chars.begin();
                p != chars.cend(); ++p)
            p->increase();
    }

    void decrease(const string &n) {
        for (auto p = chars.begin();
                p != chars.cend(); ++p)
            if (p->name() == n)
                p->decrease();
    }
```

Marking: private [2] data member [3], add [10, including 2 for const ref], increase [8], decrease [12, including 2 for const ref].

(c) We need an external function

30 marks

```
bool alive(const character &c) {
    return c.health() > 0;
}
```

Then we can define a member function

```
    int num_alive() const {
        return count_if(chars.cbegin(),
                        chars.cend(), alive);
    }
```

Marking: `alive` [15, including 2 for const and reference] `num_alive`: signature [5] (including 1 for `const`), algorithm call [10].

3. (a) The first indicates that the member function `score()` will not modify the vector passed as a parameter [5]. **10 marks**

The second indicates that `score()` will not modify any data members of the object on which it is called [5].

(b) **30 marks**

```
class average : public scorer {
public:
    double score(const vector<double> &v) const {
        double total = 0;
        for (double x : v)
            total += x;
        return total/v.size();
    }
};
```

Marking: class [5], signature [5], management of `total` [5], loop [10], return [5].

(c) **30 marks**

```
class largest : public scorer {
public:
    double score(const vector<double> &v) const {
        double max = v.front();
        for (double x : v)
            if (x > max)
                max = x;
        return max;
    }
};
```

Marking: class [5], signature [5], management of `max` [10], loop [5], return [5].

(d) **30 marks**

```
void print_scores(const scorer &sc,
        const map<string, vector<double>> &m) {
    for (const auto &p : m)
        cout << p.first << ": " <<
            sc.score(p.second) << '\n';
}
```

Marking: parameters [10], loop [10], printing [10].