Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

uOttawa

L'Université canadienne
Canada's university

CSI2372A
Advanced Programming Concepts with C++

MIDTERM EXAMINATION A

Length of Examination: 75 minutes **November 8, 2017, 14:30**

Professor: Jochen Lang **Page 1 of 10**

Family Name: _____

Other Names: _____

Student Number: _____

Signature _____

You are allowed **ONE TEXTBOOK** as a reference. No calculators or other electronic devices are allowed.

Please answer the questions in this booklet. The marks for each question are marked in the booklet with [ ].

At the end of the exam, when time is up: Stop working and turn your exam upside down. Remain silent.

Question	Marks	Maximum
A.1-A.8		8
B.1		2
B.2-3		5
B.3-4		3
B.5		4
C.1		1
C.2-3		3
Total		26

PART A: SHORT QUESTIONS (8 MARKS)

1. Clearly mark any lines causing a compile error below [1]

```
const int ci = 2;
int j = ci;
ci = j;
int i = 5;
const int cj = i;
std::cin >> ci;
```

2. Call the function `getAddIncrementCount` of `struct A` and print the return value to console, what will be printed? [1]

```
struct A {
  static int count;
  static int getAddIncrementCount() {
    return ++count;
  }
};
int A::count{0};

int main() {
  // Insert your code here









}
```

3. Convert the string "7" to the integer `i` using streams

```
string s{7};
int i;
```

4. Open the file "text.txt" for reading and print "File could not be opened" on failure.

5. Rewrite or mark up the function prototypes in the class `LetterGrade` using const and references as much as possible but not more. [1]

```
class LetterGrade {
  string d_mark{"INC"};
public:
  LetterGrade() = default;
  LetterGrade( string m )
    : d_mark(m) {}
  string get()
    { return d_mark; }
  void set(string m)
    {d_mark = m;}
  bool pass()
    { if ( d_mark < "D" || d_mark == "D+") return true; }
};
```

Consider the following definitions:

```
class Parent {
  int p{1};
public:
  virtual int getA() { return p; }
  virtual int getB()=0;
  int getC() { return p; }
 };

class Child : public Parent {
  int p{2};
public:
  int getA() { return p; }
  int getB() override { return p; }
  int getC() { return p; }
};
```

6. What is printed by the following? [1]

```
Child c;
cout << c.getA() << " " << c.getB() << " " << c.getC() << endl;
```

7. What is printed by the following? [1]

```
Child c;
Parent& p = c;
cout << p.getA() << " " << p.getB() << " " << p.getC() << endl;
```

8. Modify the program to print "Downcast failed!" on failure of the dynamic cast. [1]

```
Child c;
Parent& p = c;


auto u = dynamic_cast<Child&>(p);


 std::cerr << "Downcast failed!" << endl;
```

PART B: Programming Questions: Internal Aggregration (14 MARKS)

Consider the class definition of `StringStore` that holds strings in a growable array in the class variable `d_store`. The class is to implement internal aggregation for the growable array. Similar to the standard library, the current size of the array at `d_store` is stored in the class variable `d_capacity` while the number of strings stored is held in the class variable `d_size`.

```
class StringStore;
ostream& operator<<( ostream& os, const StringStore& stS);

class StringStore {
  std::string* d_store{0}; // pointer to array
  size_t d_size{0}; // no. of elements in array
  size_t d_capacity{0}; // size of array
public:
  StringStore() = default;
  StringStore(const StringStore& oStS );
  StringStore& operator=(const StringStore& oStS);
  ~StringStore();
  StringStore(std::vector<std::string>& sVec );
  void add(const string& str);
  friend std::ostream& operator<<( std::ostream& os,
                                   const StringStore& stS);
};
```

1. Implement the constructor `StringStore(std::vector<std::string>& sVec );` The constructor is to create an array of strings to store all the strings passed in by the sVec. Set the size and capacity of the StringStore to the number of strings in sVec. [2]

2. Implement internal aggregation for the copy constructor of `StringStore`. [2]

3. Implement internal aggregation for the assignment operator of `StringStore`. [3]

4. Implement internal aggregation for the destructor of `StringStore`. [1]

5. Implement the insertion operator to insert all strings to `ostream`, one string per line.[2]

6. Implement the function `void add(const string& str);` The function is to add the passed `str` to the `StringStore`. If the array `d_store` is full, it's capacity is to double to make room for the passed `str`. (growable array). [4]

PART C: Programming Questions: Derived Classes (4 MARKS)

Consider the class definition of `Route` that is derived from StringStore. It stores a start and destination as a string and the street names for the route in the `StringStore`.

```
class Route;
ostream& operator<<( ostream& os, const Route& rt);

class Route : protected StringStore {
   string d_destination;
   string d_start;
public:
   Route() = delete;
   Route(std::vector<string>& streetNames, string start, string
   destination);
   void add(const string& streetName );
   friend ostream& operator<<( ostream& os, const Route& rt);
};

int main() {
  std::vector<string> streetsA{"Wellington","Bronson","Laurier"};
  Route rtA{streetsA,"Home","School"};
  rtA.add( "King Edward");
  cout << rtA;
  return 0;
}
```

Program Output:

```
  From: Home to School
  Wellington
  Bronson
  Laurier
  King Edward
```

7. Is it necessary to implement the copy constructor, assignment operator and destructor for the class Route? Explain in one sentence why. [1]

8. Implement the function `void add(const string& streetName );` [1.5]

9. Implement the insertion operator for `Route`.
   `ostream& operator<<( ostream& os, const Route& rt);` [1.5]