School of Computing and Informatics

|BCS362 | TEST I | Fri 14-09-2018 |2:00-3:00PM|

1. A class is declared as

```
1 class Time
2 {
3   private:
4     int hour;
5     int min;
6     int sec;
7   public:
8     friend istream& operator>>(...);
9 };
```

   (i) Write definitions of the setHour(int), setMin(int) and setSecond(int) functions such that their calls can be cascaded. [3 Marks]

```
Time& setTime(int h, int m, int s)
{
  setHour(h);
  setMin(m);
  setSec(s);

  return *this; //enable cascading function call
}
Time& setHour(int h)
{
  if (h >= 0 && h < 24)
    hour = h;
  return *this; //enable cascading function call
}
Time& setMin(int m)
{
  if (m >= 0 && m < 60)
    min = m;
  return *this; //enable cascading function call
}
Time& setSec(int s)
{
  if (s >= 0 && s < 60)
    sec = s;
  return *this; //enable cascading function call
}
```

   (ii) Write the definition of the function declared at line 8 such that stream extraction operator can be used to read data members of **Time** from standard input device. Replace . . . with appropriate parameter list. [3 Marks]

```
istream& operator>>(istream& input, Time& t)
{
  input>>t.hour;
  input>>t.min;
  input>>t.sec;
  return input;
}
```

(iii) Write a line of code that will cascade calls to setHour(), setMin(), setSecond() in a single statement.Pass dummy parameters to each function.

[2 Marks]

```
 Time time;
 time.setHour(17).setMin(45).setSec(52);
```

2. What is data abstraction as used in object oriented programming in C++. [2 Marks]

   - *Focusing on essential/important aspects of an object and ignoring any information or aspects that are not important*

3. A program has the following lines

```
double val [10];
double *valPtr = val;
cout << valPtr << endl;
cout << val << endl;
cout << ++valPtr << endl;
cout << valPtr−− << endl;
cout << val << endl;
```

What would be the output of line 3, 5, 6, and 7 if line 4 displays **0X70FD28** on a machine that uses 64 bit hexadecimal memory addresses and 32 bits to represent doubles. [4 Marks]
Line 3 – 0X70FD28
Line 4 – 0X70FD28
Line 5 – 0X70FD30
Line 6 – 0X70FD30
Line 6 – 0X70FD28

4. Differentiate between vector capacity and size. [2 Marks]

   - Capacity is how many elements the vector can hold
   - Size is the number of elements actual held in the vector

5. A class or a function can be a friend of another class. State **THREEs** rules that restrict this kind of friendship. [3 Marks]

   - Friendship is granted
   - Friendship is not transitive
   - Friendship is not symmetric

6. A file named **input.txt** contains a line of text shown below

```
Raphael Kaka 1962 100.750 B
```

Write a complete program that will read this line of text and store it in a string **data** and consequently read the strings, integer, double and character into respective appropriately declared variables.

[3 Marks]

```cpp
#include <iostream>
#include<fstream>
#include <sstream>
using namespace std;
int main()
{
  ifstream input("input.txt");
  string data;
  getline(input, data);
  istringstream inputString {data};
  string firstname;
  string lastname;
  int year;
  double salary;
  char c;
  inputString >> firstname >> lastname >> year >> salary >> c;
  cout << firstname << "\n" << lastname << "\n" << year << "\n" << salary << "\n" << c << endl;
}
```

7. A class hierarchy is defined as

```cpp
class Person final{
  private:
    string name;
  public:
    virtual void howToMove() final {cout << "Walking"; }
}
class Student:Person{
  public:
    void howToMove() { cout << "Skiing\n"; }
    virtual Student(){ }
}
```

State and explain **THREE** errors in the code above. [3 Marks]

- Class Person is declared final hence it can not be a base class of any class but class Student is trying to inherit from it
- Function howToMove() is declared final in Person but class Student is trying to override it
- Class Student declares a virtual constructor. You can not declare a constructor as virtual

8. A class hierarchy is defined as

```cpp
class Point{
  private:
    int x, y;
  public:
};
class Line:public Point{
  private:
    int a=10, b=20;
  public:
};
```

(i) Write a directive to be included in class **Line** such that it inherits constructors of class **Point**. [2 Marks]

```cpp
    using Point::Point;
```

(ii) Assuming class **Line** has a line of code you provided as answer to (i) above, what would be the effect of the following two lines of code? [2 Marks]

```cpp
Line line(2, 4);
Line lone;
```

- Line 1 – will create an object of class **Line** and initialize members of class **Point** with values 2 and 4 respectively
- Line 2 – will throw an error since class **Line** has no default constructor

9. A class **Time** is declared as

```
#ifndef _TIME_
#define _TIME_
#include <iostream>
class Time
{
  private:
    int hour; //hold hour
    int min; //hold minutes
    int sec; //hold seconds
  public:
};
#endif
```

(i) Define setter functions for Time data members such that their calls can be cascaded. [3 Marks]

```
 Time& setHour(int h){
    if (h >= 0 && h < 24)
      hour = h;
   return *this; //enable cascading function call
 }
 Time& setMin(int m){
    if (m >= 0 && m < 60)
      min = m;
   return *this; //enable cascading function call
 }
 Time& setSec(int s){
    if (s >= 0 && s < 60)
      sec = s;
   return *this; //enable cascading function call
 }
```

(ii) Define three delegate constructors that use a fourth constructor declared as

```
Time(int, int , int);
```

to initialize variables *hour*, *min*, and *sec*. [3 Marks]

```
    Time (): Time{0, 0, 0}{}
    Time (int h) : Time{h, 0, 0}{}
    Time (int h, int m): Time{h, m, 0}{}
```

(iii) Using member initializer list, write the definition of the constructor in (ii) above. [2 Marks]

```
    Time(int hour, int m, int s): hour{hour}, min{m},sec{s}{}
```

(iv) Write the definition of a function that will return current time as a string in the form of **hour:minute:second**. [3 Marks]

```
    string toString()
    {
      ostringstream out;
      out << "Time set is: " << hour << ":" << min << ":" << sec <<endl;
      return out.str();

```

(v) Write the definition of a function declared as **void decrement()** such that it subtracts one second from the current time. [2 Marks]

```
void decrement()
{
   if(sec != 0)
     sec -= 1;
   else if (sec == 0 && min != 0){
     sec = 59;
     min -= 1;
   }
   else if(sec == 0 && min == 0){
     sec = 59;
     min = 59;
     hour -= 1;
   }
}
```

(vi) Using a member function and function defined in (v), overload prefix decrement operator so that it subtracts one second from an object of class **Time** and return a reference to new time.

[2 Marks]

```
//overload prefix --
Time& operator--()
{
  decrement(); //increment time
  return *this; // return reference to create lvalue
```

(vii) Using a non-member function and function defined in (v), overload postfix decrement operator so that it subtracts one second from an object of class **Time** and return old un-decremented time.

[2 Marks]

```
//overload postfix ++. Note, dummy int parameter has no parameter name
Time operator++(int)
{
  Time temp{*this}; //hold the current time
  increment(); //increment time
  //return unincremented, saved, temporary object
  return temp; // return a value, not a reference
```

10. A class hierarchy is defined as

```
class Shape{
  public:
    virtual double area(){return 0.0;}
    double getArea() const {return 0.0;}
};
class Rectangle:public Shape{
  private:
    int length = 10, width = 5;
  public:
    double area(){return length * width;}
    double getArea(){return length * width;}
};
```

What will be the output of the code-segment below. Explain your answers. [3 Marks]

```
Rectangle r;
Shape *s, *sh;
s = &r;
cout << s->area() << endl;
cout << s->getArea() << endl;
cout << sh->area() << endl;
```

- Line 4 – will display 50 because it is bound at runtime (dynamic binding). This calls the function in Rectangle

- Line 5 – will display 0 because it is static binding. This calls the function in class Shape
- Line 6 – will through an error because the pointer **sh** is not initialized with a reference.

11. Class Date is declared as

```
1  class Date{
2   private:
3      static const int months{ 13 };
4      static const int daysPerMonth[months];
5      static int count;
6      int day, month, year;
7   public:
8      void decrement(int);
9      bool endMonth();
10     bool leapYear();
11 };
```

(i) Write lines of code that initialize **count** to **0** and **daysPerMonth** to **{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}** at global namespace such that they are accessible to all objects of the class Date. [2 Marks]

```
int Date::count = 0;
const int Date::daysPerMonth[] = {0, 31,28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

(ii) Write the definition of the function declared at line 10 such such that it returns true if a year is not a leap year or false otherwise. A leap year is divisible by both 100 and 400 or is divisible by 4. [2 Marks]

```
    bool leapYear()
    {
      if (!((year % 400 == 0 && year % 100 == 0) || year % 4 == 0))
        return true;
      else
        return false;
    }
```

(iii) Write definition of function declared at line 9 such that it returns true if current day of the month is not the last day of the month and returns false otherwise.

[2 Marks]

```
    bool endMonth(int d)
    {
      if (month == 2 && leapYear())
        return day != 29;
      else
        return (day != daysPerMonth[month]);
    }
```

(iv) Write definition of function declared at line 8 such that it subtracts number of days passed to it from the current date. [3 Marks]

```
    void decrement(int days)
    {
      for(int i = 0; i < days; i++)
      {
        if(day == 1 && month != 1 && leapYear() == true){
          day = daysPerMonth[month-1];
          month -= 1;
        }
        else if(day == 1 && month == 3 && leapYear() == false){
          day = 29;
          month -= 1;
        }
        else  if(day == 1 && month == 1){
          day = 31;
          month = 12;
          year -= 1;
        }
        else
          day -= 1;
      }
    }
```

(v) Using function defined in (iv), overload decrement and assign operator (-=) such that it subtracts number of days it receives from the current date, and return reference of new decremented object. [3 Marks]

```
    Date& operator -=(int days)
    {
      decrement(days);
      return *this;
    }
```

12. What would be the output of the code below if a string **MMUST is a great university** is entered at line 3. [2 Marks]

```
char name[40];
char name_2[40];
cin >> name;
cout << name << endl;
cin.get(name_2, 40);
cout << name_2 << endl;
```

- Line 4 will display **MMUST**
- Line 6 will display **is a great university**

13. A function is declared as

```
int gcd(int, int);
```

(i) Write a main function that declares a constant pointer to this function. Use this pointer to invoke the function definition of this function and pass value 6 and 4 to it. [2 Marks]

```
int main()
{
  int (*const fncPtr)(int, int) = gcd;
  cout << fncPtr(6, 4) << endl;
  cout << reinterpret_cast<void*>(fncPtr) << endl;
}
```

(ii) Display the memory address where the function gcd() lives. [1 Marks]

14. A program is partially defined as

```
vector<float> data;
vector<float> num(20);
vector<float> res;
res.push_back(123);
res.push_back(54);
```

(i) Write a line of code that will preserve enough space for vector **data** such that it can contain elements of vector **num** and vector **res** even when these vectors are changing dynamically. [1 Mark]

```
data.reserve(num.size() + res.size());
```

(ii) Write a line of code that will put the value 768.125 as 15^{TH} element of vector **num**. Do not use subscript operator. [1 Mark]

```
num.insert(num.begin() + 14, 768.125);
```

(iii) Without using a loop, write two lines of code that will copy elements of vector **num** and vector **res** to vector **data**. Elements of **num** are copied to **data** before elements of **res** are copied. [2 Marks]

```
data.insert(data.end(), num.begin(), num.end());
data.insert(data.end(), res.begin(), res.end());
```