Student name: _____ **Student ID:** _____

Cairo University
Faculty of Computers and Artificial Intelligence
Subject: **Object-Oriented Programming**
Subject Code: CS213
Examiner(s): Cherry Ahmed

Mid-term exam-[A]
Semester: 1st
Date: 9/12/2020
Duration: 1 hour

/40

Question 1: Multiple Choice (20):

a. The following is(are) true about constructors:

1. Can have a return value
2. Can take parameters
3. Can be overloaded
4. both 1 and 2
5. both of 2 and 3
6. All of 1,2 and 3

b. For the following code, the value of the static variable **cnt** can be printed using:

```
class C
{
   public:
      static int cnt;
};
int C::cnt=0;
int main()
{
   C obj;
   //print cnt of class C
}
```

1. `cout<<C::cnt;`
2. `cout<<obj.cnt;`
3. `cout<<obj->cnt;`
4. both 1 and 2.
5. both of 1 and 3 .

c. Which of the following statements call the method header → **A operator++(int) {…}**

1. `A a_obj; a_obj++;`
2. `A a_obj; a_obj++(10);`
3. `A a_obj; ++a_obj;`
4. None of the above

d. Which of the following is true about ***struct*** and ***class***:

1. struct members are private by default, while class members are public by default
2. struct members are public by default, while class members are private by default
3. both of struct and class members are private by default
4. both of struct and class members are public by default

e. To be able to set and get array elements of the class **ClassRoom,** the subscript operator [ ] was overloaded. The datatypes of the overloaded function's **return** and **parameter** should be:

```
class ClassRoom
{
   Student* students;
   int size;
   public:
      ClassRoom(int s):size(s)
      {
         students=new Student[size];
      }
       ----- operator[](----- var);
};
```

1. int, Student
2. int &, Student
3. Student &, int
4. Student, int
5. Student&, Student
6. int, Student&

f. Assume class TEST. Which of the following statements is/are responsible to invoke a copy constructor?

1. TEST T2(T1)
2. TEST T4 = T1
3. T2 = T1
4. both 2 and 3
5. both of 1 and 2
6. All of 1,2 and 3 invoke the copy constructor

g. Assuming the following code:

```
class P
{
  int x;
  public:
        _ _ _ _ fun()
      { return this;}
};
```

The member function ***fun*** should have a return type:

1. void
2. P&
3. P*
4. int

h. Assume you overload a relational operator in a class named "**Cat**", the return of the overloaded function should be of type:

1. Cat
2. int
3. bool
4. ostream&
5. istream&

i. Which of the following is the correct syntax to call a member function using a pointer?

1. pointer->function()
2. pointer.function()
3. pointer::function()
4. pointer:function()
5. &pointer->function

j. What is the output of the following?

```
class A
{
     int x;
     public:
       A(int val) { x= val;}
       void setX(int val) { x = val;}
       int getX() { return x; }
};
```

```
int main()
{
      A obj;
      obj.setX(7);
      cout<<"X="<<obj.getX();
      return 0;
}
```

1. X= 7
2. X= 0
3. Runs but no output
4. Compiler error

MCQ Answer: each 2 marks

a	b	c	d	e	f	g	h	i	j
5	4	1	2	3	5	3	3	1	4

Question 2: Answer the following (6):

```
class House
{
    string city;
    int rooms;
    int area;
public:
    void setVals
(string c, int r,int a)
    {
        city = c;
        rooms = r;
        area = a;
    }

friend class Person; [1]
};
```

```
class Person
{
    string name;
    string job;
    House home;
public:
    Person(string n,string j, House h)//constructor 1
    {
        name = n;
        job = j;
        home = h;
    }

Person(string,string,string,int,int);//constructor 2
};
```

a. Add 1 line to class **House** such that class **Person** can access private members of House.

b. Implement the second constructor of class Person. It takes the following parameters: **name** and **job** of a person, and **city**, **rooms** and **area** of the person's home.

```
Person :: Person(string n,string j,string c,int r,int a){
    name=n; [0.5]
  job=j;  [0.5]
  home.city=c; [0.5]
  home.rooms=r;  [0.5]
  home.area = a;}  [0.5]
```

Or replace last 3 lines with:

```
home.setVals(c,r,a); [1.5]
```

c. Add a **main function** that creates 2 Person objects using a different constructor for each object.

```
int main()
{
  Person p("Hassan","Worker","Cairo",2,100); [1]
  House h; [0.5]
  h.setVals("Alex",3,140); [0.5]
  Person p2("Mohamed","Doctor",h); [0.5]
  return 0;
}
```

Question 3: Answer the following (14):

```
int main()
{
    MyTime t1(20,3);
    t1.display(); //displays 20:03

    MyTime t2(100);
    t2.display(); //displays 01:40

    MyTime t3=t1++;
    t3.display(); //displays 20:03
    t1.display(); //displays 20:04

    MyTime t4= t3-10;
    t4.display(); //displays 19:53
    return 0;
```

Write a class **MyTime** that stores a time of the day as integer private members **hours** and **minutes**. The class takes the hours in a 24-hour format, which means that the hour can take the values 0-23. So the time 20:15 is equivalent to 8:15 pm.
The class has **3 constructors**:
1- The first takes hours and minutes as parameters.
2- The second takes minutes only and converts them to hours and minutes.
3- The third is a copy constructor.
The class **overloads:**
4- The **++ postfix operator** to increment minutes
5- And the **– operator** to subtract minutes from a time.
The class also has:
6- A **display** method that prints the time in the format “hh:mm”.

The main function on the right should work

```
}
```

```
class MyTime

{
    int hours, minutes; [1]
public:
    MyTime(int,int);
    MyTime(int);
    MyTime(const MyTime&);
    MyTime operator++(int);
    MyTime operator-(int);
    void display();
};
void MyTime::display()
{
    cout<<endl;
    if(hours<10)
        cout<<"0";[0.5]
    cout<<hours<<":"; [0.5]
    if(minutes<10)
        cout<<"0";[0.5]
    cout<<minutes; [0.5]
}
MyTime::MyTime(int h,int m)
{
    hours = h; [0.5]
    minutes = m; [0.5]
}
MyTime::MyTime(int m) [0.5]
{
    minutes = m%60; [0.5]
    hours = m/60; [0.5]
}
```

```
MyTime::MyTime(const MyTime& obj) [0.5]
{
    hours = obj.hours; [0.5]
    minutes = obj.minutes; [0.5]
}


MyTime MyTime::operator++(int) [0.5]
{
    MyTime tmp(*this); [0.5]
    minutes++; [0.5]
    if(minutes>=60) [0.5]
    {
        minutes = 0; [0.5]
        hours++;[0.5]
       if(hours>23)
            hours-=24; [0.5]
    }
    return tmp; [0.5]
}
MyTime MyTime::operator-(int mins) [0.5]
{
    minutes-=mins; [0.5]
    if(minutes<0) [0.5]
    {
        minutes +=60; [0.5]
        hours--;[0.5]
        if(hours<0)
            hours+=24; [0.5]
    }
    return *this; [0.5]
}
```