Student name: ____ **Student ID:** ________________

Cairo University
Faculty of Computers and Artificial Intelligence
Subject: Object-Oriented Programming
Subject Code: CS213

Mid-term exam-B
Semester: 1st
Date: 29/11/2021
Duration: 1 hour

/40
/20

Question 1: Multiple Choice (20):

1. Which member function of class cannot modify its objects attributes?
- **a)** Inline member functions
- **b)** Private member functions
- **c)** Static member functions
- [✓ **d)** Constant member functions]

2. If a default constructor is not defined, then how the objects of the class will be created?
- **a)** The compiler will generate error
- **b)** Error will occur at run-time.
- [✓ **c)** Compiler provides its default constructor to build the object]
- **d)** None of the above

3. Which of the following calls the copy constructor of the class Person:
- **a)** Person s1(4,5); Person s2(s1);
- **b)** Person s1(4,5); Person s2 = s1;
- **c)** Person s1(4,5); Person s2; s2 = s1;
- [✓ **d)** Both of a and b are correct.]
- **e)** Both of a and c are correct.
- **f)** All of a , b and c are correct.

4. To <u>initialize</u> the static int array ***years*** of class Student, the following statement should be used:
- [✓ **a)** int Student::years[4]={1,2,3,4};]
- **b)** static int Student::years[4]={1,2,3,4};
- **c)** static int years[4]={1,2,3,4};
- **d)** int Student.years[4] = {1,2,3,4};

5. The header of the operator== of the class Person should be:
- [✓ **a)** bool Person::operator==(const Person&p)]
- **b)** void Person::operator==(const Person&p1 , const Person&p2)
- **c)** Person Person::operator==(const Person&p)
- **d)** bool Person::operator==(const Person&p1, const Person&p2)

6. If a class has a character array as an attribute, and has overloaded the subscript [ ] operator to set and get the characters of the array. The prototype of the overloaded operator should be:
- **a)** char operator[ ](int);
- [✓ **b)** char& operator[ ](int);]
- **c)** int& operator[ ](char);
- **d)** int& operator[ ](int);

Assuming the following code:

```
class MyTime
{
  int hr, min;
  public:
   MyTime(int h,int m)
   { set(h,m);}
   void set(int h,int m) {
     min = m;
     hr = h;           }
   void display() {
    cout<<hr<< ": "
    <<min<<endl;
                  }
   MyTime operator-(int);
};
```

```
class TimeDuration
{
   MyTime start_time,end_time;
   string desc;
   public:
    void display() {
    cout<<"Description:"<<desc<<endl;
    cout<<"Start Time:";
    start_time.display();
    cout<<"End Time:";
    end_time.display();
                        }
};
```

7. There is a problem with the previous code that will appear once an object of class **TimeDuration** is created, which of the following is it?

a) A missing copy constructor in TimeDuration **b)** A missing copy constructor in MyTime
c) A missing default constructor in TimeDuration [✓ **d)**] A missing default constructor in MyTime

8. For the previous code, a parameterized constructor for the class **TimeDuration** might have the prototype:

a) `TimeDuration(MyTime,MyTime,string)` **b)** `TimeDuration(int,int,int,int,string)`
c) `TimeDuration(string,int,int,int,int)` [✓ **d)**] All of the mentioned are valid

9. For the previous code, which of the following statements might appear in the parameterized constructor of TimeDuration?

a) `start_time.hr = H;` **b)** `start_time->set(H,M);`
[✓ **c)**] `start_time = obj;` **d)** `start_time.hr = obj.hr;`

10. Which of the following is a possible way to use the operator – in class MyTime? (given that t1 and t2 are objects of class MyTime)

[✓ **a)**] `MyTime result = t2-10-13;` **b)** `int result = t2-10;`
c) `MyTime result = t2-t1;` **d)** `MyTime* result = t2-10;`

Question 2: Answer the following? [20 marks]

Declare and implement a class **Encryptor** that stores a dynamic character array (**sentence**) and its size (**_size**) **(1 mark)**. The class is used to store a sentence and provides operators += and -= to encrypt and decrypt the sentence.

The class should have:

A. A constructor that takes the size of the array as a parameter. **(2 marks)**

B. A copy constructor **(3 marks)**

C. A member function setSentence that sets the member character array to the parameter array. The function returns a Boolean in case the parameter array has a length greater than the size of the array. **(3 mark)**

D. Overload the operator += which uses an integer value added to each character of the array to encrypt. For example, to encrypt the sentence of the object encObj by the integer value 3, the statement (encObj += 3) is used. If the sentence is “Hi there”, it will be changed to “Kl#wkhuh” **(3 marks)**

E. Overload the operator -= which uses an integer value subtracted from each character of the array to decrypt. For example, to decrypt the sentence of the object encObj by the integer value 3, the statement (encObj -= 3) is used. If the sentence is “Kl#wkhuh”, it will be changed to “Hi there” **(3 marks)**

F. Overload the insertion operator<< so that it prints the sentence of an Encryptor object. **(3 marks)**

G. Your class should include a destructor. **(1 marks)**

Note: You may use the functions strlen and strcpy from the cstring library.

```
int main()
{
     Encryptor e(20);
    if(!e.setSentence("Good night"))
    {
        cout<<"Sentence too big";
        return 0;
    }
    cout<<"Original Sentence:";
    cout<<e;//prints "Good night"
    e+=3;
    cout<<"Encrypted Sentence:";
    cout<<e;//prints "Jrrg#qljkw"
    e-=3;
    cout<<"Decrypted Sentence:";
    cout<<e;//prints "Good night"
    Encryptor e2 = e+=4;
    //prints "Original:Kssh$rmklx"
    cout<<"Original:"<<e;
    //prints "Copy:Kssh$rmklx"
    cout<<"Copy:"<<e2;
    return 0;
}
```

Answer:

```
class Encryptor
{
   char * sentence;
   int s;
   public:
      Encryptor(int _s)
      {
         s = _s;
         sentence = new char[s];
      }
      Encryptor(const Encryptor& obj)
      {
         s = obj.s;
         sentence = new char[s];
         strcpy(sentence,obj.sentence);
      }
      bool setSentence(char* str)
      {
         if(strlen(str) >= s)
            return false;
         else
         {
            strcpy(sentence,str);
            return true;
         }
      }
      friend ostream& operator<<(ostream& os,const Encryptor& e)
      {
         os<<e.sentence<<endl;
         return os;
      }
      Encryptor operator+=(int key)
      {
         for(int i=0;i<strlen(sentence);i++)
            sentence[i] += key;
         return *this;
      }
      Encryptor operator-=(int key)
      {
         for(int i=0;i<strlen(sentence);i++)
            sentence[i] -= key;
         return *this;
      }
      ~Encryptor(){delete[] sentence;}
};
```