

جامعة بنها - كلية العلوم - قسم الرياضيات
مادة من المستوى الثاني (علوم حاسب- رياضيات-فيزياء)

يوم الامتحان: الخميس

تاريخ الامتحان: 2 / 1 / 2020 م

المادة : برمجة شينية (251 رس)

الممتحن: د/ مصعب عبد الحميد محمد حسان

مدرس بقسم الرياضيات بكلية العلوم

نموذج الإجابة

ورقة كاملة



Object Oriented Programming (MC251) for Second Level Students

Choose the correct answer for each of the following[15x2+6x3 = 48]:

- 1- A class is an expanded concept of a data structure. it can hold
(A) data (B) functions (C) both data and functions
- 2- members are accessible from members of their same class and from their friends, but also from members of their derived classes.
(A) Public (B) Protected (C) Private
- 3- members are accessible from anywhere where the object is visible.
(A) Public (B) Protected (C) Private
- 4- The function operator+ of an class is the one that is in charge of overloading the addition operator (+). This function can be called either implicitly using the operator, or explicitly using the function name:
c = a + b;
c = a.operator+ (b);
Both expressions are equivalent.
(A) Yes (B) No
- 5-The is automatically called when an object is destroyed.
(A) Class (B) Constructor (C) Destructor (D) New
- 6- The keyword represents a pointer to the object whose member function is being executed. It is a pointer to the object itself.
(A) class (B) operator (C) this
- 7- A constructor can also be overloaded with more than one function that have the same name but different types or number of parameters.
(A) Yes (B) No
- 8- If you do not declare any constructors in a class definition, the compiler assumes the class to have a default constructor with
(A) one argument (B) two arguments (C) three arguments (D) no arguments
- 9- of a class are also known as "class variables", because there is only one unique value for all the objects of that same class.
(A) Data members (B) Static data members (C) Function members
- 10- If we want to declare an external function as friend of a class, thus allowing this function to have access to of this class.
(A) the private and protected members (B) the private and public members
(C) the private members (D) the public members
- 11- Inheritance allows to create classes which are derived from other
(A) classes (B) data members (C) function members
- 12- In C++ it is perfectly possible that a class inherits members from
(A) one class (B) more than one class
- 13- If we want to define a different implementation for a template when a specific type is passed as template parameter, we can declare
(A) the same template (B) a specialization of that template.

What will be the output of the following C++ programs?

14- #include <iostream.h>

```
class CRectangle {
    int x, y;
public:
    void set_values (int,int);
    int area () {return (x*y);}
};
void CRectangle::set_values (int a, int b) {
    x = a;      y = b;}
int main () {
    CRectangle rect;
    rect.set_values (5,7);
    cout << rect.area();
    system("pause"); return 0;}
```

(A) 25 (B) 49 (C) 35 (D) 12

15- #include <iostream.h>

```
class CDummy {
public: static int n;   CDummy () { n++; };   ~CDummy () { n--; };
};
int CDummy::n=0;
int main () {
    CDummy a;   CDummy b[10];
    CDummy * c = new CDummy;
    cout << a.n << " ";
    delete c;
    cout << CDummy::n << endl;
    system("pause"); return 0;}
```

(A) 10 11 (B) 11 10 (C) 11 12 (D) 12 11

16- #include <iostream.h>

```
class mother {
public:
    mother () { cout << "mother: no parameters\n"; }
    mother (int a) { cout << "mother: int parameter\n"; }
};
class daughter : public mother {
public:
    daughter (int a) { cout << "daughter: int parameter\n\n"; }
};
class son : public mother {
public:
    son (int a) : mother (a)
    { cout << "son: int parameter\n\n"; } };
int main () {
    daughter A(0);
    son B(0);
    system("pause"); return 0;}
```

(A)	mother: no parameters daughter: int parameter mother: int parameter son: int parameter	(B)	mother: int parameter daughter: int parameter mother: int parameter son: int parameter
-----	---	-----	---

```

17- #include <iostream.h>
    template <class T>
    class mycontainer {
        T element;
        public: mycontainer (T arg) {element=arg;}
               T FUN1 () {return element;}
    };
    template <>
    class mycontainer <char> {
        char element;
        public: mycontainer (char arg) {element=arg;}
               char FUN2 (){
                    if ((element>='a')&&(element<='z'))
                        element+='A'-'a';
                    return element;}
    };
    int main () {
        mycontainer<int> myint (7);
        mycontainer<char> mychar ('j');
        cout << myint.FUN1() << " ";
        cout << mychar.FUN2() << endl;
        system("pause"); return 0;}
    (A) 7 j      (B) 8 j      (C) 7 J      (D) 8 J

```

```

18- #include <iostream.h>
    class CVector {
        public:
            int x,y;
            CVector () {};
            CVector (int,int);
            CVector operator+ (CVector);
    };
    CVector::CVector (int a, int b) {
        x = a;
        y = b;
    }
    CVector CVector::operator+ (CVector param) {
        CVector temp;
        temp.x = x * param.x;
        temp.y = y - param.y;
        return (temp);
    }
    int main () {
        CVector a (3,1);
        CVector b (1,2);
        CVector c;
        c = a + b;
        cout << c.x << " " << c.y;
        system("pause"); return 0;}

```

(A) 4 3 (B) 2 -1 (C) 1 6 (D) 3 -1

```

19- #include <iostream.h>
template <class T, int N>
class mysequence {
    T memblock [N];
public:
    void setmember (int x, T value);
    T getmember (int x);};
template <class T, int N>
void mysequence<T,N>::setmember (int x, T value){ memblock[x]=value;}
template <class T, int N>
T mysequence<T,N>::getmember (int x) {
    return memblock[x];}
int main () {
    mysequence <int,5> myints;
    mysequence <double,5> myfloats;
    myints.setmember (2, 10);    myfloats.setmember (3, 0.1);
    cout << myints.getmember(2) + myfloats.getmember(3) << '\n';
    system("pause"); return 0;}
(A) 5          (B) 2.1          (C) 10.3          (D) 10.1

```

```

20- #include<iostream.h>
class Base {
public:
    Base(){cout << "Base" << " ";}
    Base(int i){ cout << "Base" << i << " "; }
    ~Base(){ cout << "Destruct Base" <<endl; } };
class Der: public Base{
public:
    Der(){ cout << "Der" << " "; }
    Der(int i): Base(i) { cout << "Der" << i << " "; }
    ~Der(){ cout << "Destruct Der" << endl; } };
int main(){
    Base a;    Der d(2);    system("pause"); return 0;}
(A) Base Base2 Der2    (B) Base Base Der    (C) Base2 Base2 Der2

```

```

21- #include <iostream.h>
#include <string.h>
class String{
private: char str[80];
public: String() { strcpy(str, ""); }
    String( char s[] ) { strcpy(str, s); }
    void display() { cout << str; }
    void getstr() { cin.get(str, 80); }
    bool operator == (String ss){return ( strcmp(str, ss.str)==0 ) ? true : false;};}
int main(){
    String s1 = "yes";    String s2 = "no";    String s3 = "no";
    if(s3==s1) cout << "You typed yes \n ";
    else if(s3==s2)
        cout << "You typed no \n";
    else
        cout << "You didn't follow instructions \n";
    system("pause"); return 0;}
(A) You typed yes          (B) You typed no          (C) You didn't follow
instructions

```

Model Answer

- 1- C
- 2- B
- 3- A
- 4- A
- 5- C
- 6- C
- 7- A
- 8- D
- 9- B
- 10- A
- 11- A
- 12- B
- 13- B
- 14- C
- 15- D
- 16- A
- 17- C
- 18- D
- 19- D
- 20- A
- 21- B