ESERCIZI C++ 2015/2016

N. De Francesco

1. Indicare l'uscita del seguente programma. Spiegare le eventuali differenze.
   a) Così come è scritto
   b) Senza le istruzioni indicate da *

```
class alpha {
protected:
    int a;
public:
    alpha(){a=10; cout << a << "nuovo 5  " <<  endl;};
    void g() {cout <<  a+1 << endl; }
};

class beta: public alpha {
*   protected:
*   int a;
public:
    beta() {a=11; cout << a << "nuovo 6  " << endl;}
    void    g() {cout << a+2  << endl;}
};

template<class tipo>
void funzione( tipo *obj){
    obj->g();
};
void main(){
    beta *obj1= new beta;
    alpha *obj2=obj1;
    funzione(obj1);
              funzione(obj2);
    }
```

2.

   a) Indicare l'uscita del programma seguente:

   b) Dire quali istanze delle classi template e funzioni template sono generate dal compilatore

```
#include<iostream>
using namespace std;

template<class tipo>
class uno {
        tipo a;
public:
        uno(tipo x){ a=x;}
        tipo valore(){ return a;}
};

template<class tipo, class tipo1>
void f(uno<tipo>& obj, tipo& y, tipo1 z){

         y=obj.valore();
         cout << z;}


int main() {
        int h; char c;
        uno<int> obj1(5);
        uno<char>obj2('m');
        f(obj1, h, 10);
        f(obj2, c, 2.5);
```

```
        cout << h << c;
}
```

3. Indicare l'uscita del programma seguente.

```
class A{
protected:

int a;

public:
A(){ a=10; cout << a  << endl; }
A (int x) { a=x; cout << a << endl; }
void f(){cout << a  << endl; }
};

class B : public A{
public:
B(){cout << a  << endl; }
B (int x) : A(3) { a=x; cout << a << endl; }
};

class C : public A{
public:
C(){a= 30; cout << a  << endl; }
void virtual f(){cout << a  << endl;  };
};

class D : public C {
public:
D(){a= 20; cout << a  << endl; }
void f(){cout << a + 20  << endl;  }
};

int main (){
B* objb= new B(40);
objb->f();

D* objd=new D;
A* obja=objd;
C* objc=objd;

obja->f();
objc->f();
}
```

4. Indicare l'uscita del seguente programma

```
class uno {
         public:
        uno() {}
        void   f() {
                cout << "f chiamata da uno" << endl;
        }
};
class due : public uno{
public:
        due() {}
        void virtual  f() {
                cout << "f chiamata da due" << endl;
        }
};
class tre: public due{
public:
        tre() {}
        void f() {
                cout << "f chiamata da tre" << endl;
        }
};
void main(){
        uno obj1;
        due obj2;
        tre obj3;
```

```
        obj1.f();
        obj2.f();
        obj3.f();
        uno* punt12= &obj2;
        uno* punt13= &obj3;
        due* punt23= &obj3;
        punt12->f();
        punt13->f();
        punt23->f();
}
```

5. Indicare l'uscita del seguente programma con

a) input 0

b) input 1

```
class uno {
protected:
        int a;
public:
        uno(){a=2; cout << a << '\t';}
        ~uno() { cout << a << '\t';}

};
class due : public uno{
public:
        due() {a=4; cout << a << '\t';}
due(int z) {a=z; cout << a << '\t';}
        ~due() { cout << a << '\t';}
};
class tre : public due{
protected:
        int a;
        int b;
public:
        tre(){b=6; a=9; cout << a << '\t';}
tre(int y): due (y) {b=6; a=9+y; cout << a << '\t';}
        ~tre() { cout << a << '\t';}
};
void main(){
        int x;
        cin >> x;
        if (x)  tre obj (x); else due obj;
}
```

6. Indicare l'uscita del seguente programma : a) così come è scritto; b) togliendo la funzione stampa dalla classe B. Indicare le funzioni che vengono chiamate e spiegare il risultato dei due programmi.

```
class A{
        protected:
               int x;
        int y;
public:
        A(){ x=1, y=2; };
        void stampa () {cout << x << '\t' << y << endl; };
};
class B: public A
{
    int y;
    public:
    B() { x=3; y=4; };
void stampa () {cout << x << '\t' << y << endl; };
};

int main(){
B* obj = new B;
obj->stampa();
}
```

7. Indicare l'uscita del seguente programma.

```
template<class tipo>
class uno {
    int w;
       static int s;
public:
     uno() { w=12; }
       int cambia() {w++; cout << w << "\t" ;
return  s+=10; }
};

template<class tipo>
int uno<tipo>::s;



int main() {
       uno<int> obj1, obj2;
       uno<char> obj3;
       cout << obj1.cambia() << endl;
       cout << obj2.cambia() << endl;
       cout << obj3.cambia() << endl;
       cout << obj1.cambia() << endl;
       }
```