quiz2. Code=1 Digipen login:_____

1. **Problem** (6 * 2 pts):

   Given the following definitions

```
class B1 {
      int i;
    public:
        B1() : i(1) {}
        virtual int foo() { return i; }
};
class D1 : public B1 {
      int j;
    public:
        D1() : j(2) {}
        int foo() { return j; }
};

class B2 {
      int i;
    public:
        B2() : i(3) {}
        int foo() { return i; }
};
class D2 : public B2 {
      int j;
    public:
        D2() : j(4) {}
        int foo() { return j; }
};

int f1() {
    B1* p = new D1;
    return p->foo();
}

int f2() {
    B2* p = new D2;
    return p->foo();
}

int f3() {
    B1* p = new B1;
    return static_cast<D1*>(p)->foo();
}

int f4() {
    B2* p = new B2;
    return static_cast<D2*>(p)->foo();
}

int f5() {
    B2* p = new D2;
    return dynamic_cast<D2*>(p)->foo();
}

int f6() {
    B1* p = new B1;
    if ( D1* pd = dynamic_cast<D1*>(p) ) return pd->foo(); else return 0;
}
```

What happens in each of the following function calls (each function is compiled and ran separately):

A) compiles and returns 1 B) compiles and returns 3 C) compiles and returns 2 D) compiles and returns 0 E) compiles and returns 4 F) compiles, but run-time error G) does not compile	1-1._____ `f1();` 1-2._____ `f2();` 1-3._____ `f3();` 1-4._____ `f4();` 1-5._____ `f5();` 1-6._____ `f6();`

2. **Problem** (4 * 3 pts):

What does this program output?

```
class B {
  public:
    int b;
    B(int _b=0) {
        b=_b;
        std::cout << "B("<< b <<")" << std::endl;
    }
    // not virtual !!!
    ~B() { std::cout << "~B("<< b <<")" << std::endl; }
};

class D : public B {
  public:
    int d;
    D(int _b=0, int _d=0) {
      b=_b;
      d=_d;
      std::cout << "D("<< _b <<","<< d <<")" << std::endl;
    }
    ~D() { std::cout << "~D("<< b <<","<< d <<")" << std::endl; }
};

int main() {
    //PART 1
    B** arr = new B*[2];

    //PART 2
    arr[0] = new B(1);
    arr[1] = new D(2,5);

    //PART 3
    delete arr[0];
    delete arr[1];

    //PART 4
    delete [] arr;
}
```

A) `B(1) B(2) D(2,5 )` B) `~B(0) ~B(0)` C) `B(1) D(2,5 )` D) `B(1) B(0) D(2,5 )` E) no output F) `B(1) D(2,5 ) B(2)` G) `~B(1) ~B(2)` H) `~B(1) ~B(2) ~D(2,5)` I) `~B(1) ~D(2,5) ~B(2)`	2-1._____ Part 1 2-2._____ Part 2 2-3._____ Part 3 2-4._____ Part 4

3. **Problem** (10 * 1 pts):

Given the following definitions

```
class B {
  public:
    virtual void f1();
             void f2();
};

class D : public B {
  public:
    void f1();
    void f2();
    void f3();
};

B b;
D d;
B *pb1 = &b, *pb2 = &d;
```

corresponding function of which class is called for each of the following statement, choose NC if does not compile.

A) `D::fx()` B) `B::fx()` C) `NC`	3-1._____ `b.f1();` 3-2._____ `b.f2();` 3-3._____ `pb1->f2();` 3-4._____ `b.f3();` 3-5._____ `d.f3();` 3-6._____ `pb1->f1();` 3-7._____ `pb1->f3();` 3-8._____ `pb2->f1();` 3-9._____ `pb2->f2();` 3-10._____ `pb2->f3();`

4. **Problem** (7 * 2 pts):

Which of 7 methods below compile? Notice that they are all syntactically sound.

```
class C {
  public:
    C() { data = new int (100); }
    ~C() { delete data; }
    int        GetInt()         const { return *data; }
    int&       GetRefInt()      const { return *data; }
    const int& GetRefConstInt() const { return *data; }

    int*         GetPtr()         const { return data; }
    int*&        GetRefPtr()      const { return data; }
    const int*&  GetRefPtrConst() const { return data; }
    int* const & GetRefConstPtr() const { return data; }
  private:
    int * data;
};
```

A) fails B) compiles	4-1._____ GetInt() 4-2._____ GetRefInt() 4-3._____ GetRefConstInt() 4-4._____ GetPtr() 4-5._____ GetRefPtr() 4-6._____ GetRefPtrConst() 4-7._____ GetRefConstPtr()

5. **Problem** (2 * 4 pts):

Given these classes

```
class B {
  public: virtual std::string name() { return "B"; }
};

class D : public B {
  public: virtual std::string name() { return "D"; }
};
```

what is the output of each of the following `mains`? Notice that each `main` uses a different function `foo`.

A) In foo: D In main: B

B) In foo: B In main: D

C) In foo: D In main: D

D) In foo: B In main: B

5-1._____

```
B foo(B& b) { std::cout << "In foo: " << b.name(); return b; }

int main() {
  D d;
  std::cout << "In main: " << foo(d).name();
}
```

5-2._____

```
B foo(B b) { std::cout << "In foo: " << b.name(); return b; }

int main() {
  D d;
  std::cout << "In main: " << foo(d).name();
}
```

6. **Problem** (7 * 2 pts):

Given the following definitions

```
class B            { public: int i; };
class D : public B { public: int j; };
int foo0(B  b) { return b.j;  }
int foo1(B  b) { return static_cast<D*>(&b)->j;  }
int foo2(B& b) { return static_cast<D*>(&b)->j;  }
int foo3(B* pb) { return static_cast<D*>(pb)->j; }
int foo4(B* pb) { return dynamic_cast<D*>(pb)->j; }

B base;
base.i = 1;
D derived;
derived.i=1;
derived.j=2;
```

What happens in each of the following cases:

A) All good B) compiles, but run-time error C) does not compile	6-1._____ `foo0(derived);` 6-2._____ `foo1(derived);` 6-3._____ `foo2(base);` 6-4._____ `foo2(derived);` 6-5._____ `foo3(&base);` 6-6._____ `foo3(&derived);` 6-7._____ `foo4(&derived);`

7. **Problem** (10 * 1 pts):

Given the following definitions

```
+--------------------------+------------------------------+
| class B {                |     class D : public B {     |
|   public:                |       public:                |
|     virtual void f1();   |         void f1();           |
|              void f2();  |         void f2();           |
|              void f4();  | virtual void f4();           |
| };                       |         void f3();           |
|                          |     };                       |
+--------------------------+------------------------------+

B base;
D derived;
B *pb1 = &base, *pb2 = &derived;
```

corresponding function of which class is called for each of the following statement, choose NC if does not compile.

A) `B::fx()` B) `D::fx()` C) `NC`	7-1._____ `b.f2();` 7-2._____ `pb1->f1();` 7-3._____ `pb1->f2();` 7-4._____ `pb1->f3();` 7-5._____ `pb2->f1();` 7-6._____ `pb2->f2();` 7-7._____ `pb2->f3();` 7-8._____ `d.f3();` 7-9._____ `*pb1 = *pb2; pb1->f1();` 7-10._____ `pb2->f4();`

8. **Problem** (3 * 2 pts):

Determine which of the following definitions are legal:

A) illegal

B) legal

8-1._____ `template <typename T> void func(T x=10) {}`

8-2._____ `template <int T> void func(int t=T) {}`

8-3._____ `template <int T=10> void func(int t) {}`

9. **Problem** (6 * 2 pts):

Given the following definitions

```
template <typename T1, typename T2> bool compare(T1 lhs,T2 rhs)     {cout<<"1";}
template <>                         bool compare(int lhs, int rhs)  {cout<<"2";}

template <typename T1, typename T2> bool compare(T1 * lhs,T2 * rhs) {cout<<"3";}
template <>                         bool compare(const char * lhs, const char * rhs) {cout<<"4";}

bool compare (int lhs, int rhs)            {cout<<"5";}

const char * str1 = "A";
const char * str2 = "B";
double d1=1.0, d2=2.0;
int i1=1,i2=2;
```

what is printed in each of the following cases?

A) does not compile B) 3 C) 4 D) 2 E) 1 F) 5

9-1._____ `compare(i1,i2);`

9-2._____ `compare(str1,i2);`

9-3._____ `compare(&i1,&i2);`

9-4._____ `compare<int,int>(i1,i2);`

9-5._____ `compare<int,int>(d1,d2);`

9-6._____ `compare(str1,str2);`

10. **Problem** (5 * 2 pts):

Given the definitions:

```
template <typename T> void foo(T a)       { cout << "1"; }
template <>           void foo(int  a)    { cout << "2"; }

template <typename T> void foo(T* a)      { cout << "3"; }
template <>           void foo(int* a)    { cout << "4"; }
template <>           void foo(double* a) { cout << "5"; }

                      void foo(int* a)    { cout << "6"; }

double d=1.0; int i=7; char ch='a';
```

What is printed for each of the following, choose "does not compile" if code is illegal?

A) 1 B) 3 C) 2 D) 5 E) 6 F) 4 G) does not compile	10-1._____ `foo(d);` 10-2._____ `foo(&i);` 10-3._____ `foo(&d);` 10-4._____ `foo<int>(&i);` 10-5._____ `foo<double>(d);`

11. **Problem** (7 * 2 pts):

Given the definitions and paragraph from C++ standard:

```
/*
 * 14.8.2.1 Deducing template arguments from a function call [temp.deduct.call]
 *
 * Template argument deduction is done by comparing each function template parameter
 * type (call it P) with the type of the corresponding argument of the call (call it A)
 * as described below.
 *
 * If P is not a reference type:
 * -- If A is an array type, the pointer type produced by the array-to-pointer
 *     standard conversion (4.2) is used in place of A for type deduction; otherwise,
 * -- If A is a cv-qualified type, the top level cv-qualifiers of A's type are
 *    ignored for type deduction.
 */

template <typename T> void fooRef(T& a)   { }
template <typename T> void fooVal(T  a)   { }
template <typename T> void fooPtr(T* arg) { }

int a [] = {1,2,3,4,5};
const int ca [] = {1,2,3,4,5};
int i = 10;
const int ci = 100;
int * pi = &i;
const int * pci = &i; // Pointer to Constant Int
const int * const cpci = &i; // Constant Pointer to Constant Int
int & ri = i;
const int & rci = ci; // Reference to Constant Int
```

Determine what type compiler chooses for parameter T. Choose "does not compile" if code is illegal?

A) `int` B) `const int` C) `const int *` D) `const int * const` E) `const int &` F) `does not compile` G) `int*` H) `const int [5]` I) `int * const` J) `int &` K) `int [5]`	11-1._____ `fooVal(ca);` 11-2._____ `fooRef(a);` 11-3._____ `fooVal(ci);` 11-4._____ `fooRef(ci);` 11-5._____ `fooVal(&ci);` 11-6._____ `fooPtr(cpci);` 11-7._____ `fooVal<int&>(ci);`

12. **Problem** (8 * 1 pts):

Given the three classes defined below, answer whether each of the following statements compiles or not:

```
template <typename T1 = int,
                int T2 = 10>
class Bar {
  public:
    Bar(int x = 0) { }
  private:
    T1 Do(T1 arg) {
      return arg;
    }
};
```

```
class A {
  public:
    A() { }
  private:
    //PRIVATE COPY!!!
    A(const A&) {}
};
```

```
class B {
  public:
    B(int x) : x_(x) { }
    operator int(void) {
      return x_;
    }
  private:
    int x_;
};
```

A) Does not compile B) Compiles

12-1._____ `Bar<int, 5> bar1;`

12-2._____ `Bar<int, B(5)> bar3;`

12-3._____ `Bar<A> bar4(B(5));`

12-4._____ `Bar<B, 5> bar6(5);`

12-5._____ `Bar<> bar8;`

12-6._____ `Bar<5> bar9;`

12-7._____ `int size = 8; Bar<int, size> bar11;`

12-8._____ `Bar<B(5)> bar12;`

13. **Problem** (4 * 2 pts):

Given

```
template <typename T, typename U = typename T::V >
class A { };

class B { public : typedef int V; };
```

determine which of the following declarations are legal:

A) legal B) illegal	13-1._____ `A<int> a1;` 13-2._____ `A<int,int> a2;` 13-3._____ `A<B> a3;` 13-4._____ `A<B,B> a4;`

Copy your multiple choice answers from exam into the table below
Detach this page and take it with you.
Submit multiple choice answers on pontus:

Code = 1

Q1_1	
Q1_2	
Q1_3	
Q1_4	
Q1_5	
Q1_6	
Q2_1	
Q2_2	
Q2_3	
Q2_4	
Q3_1	
Q3_2	
Q3_3	
Q3_4	
Q3_5	
Q3_6	
Q3_7	
Q3_8	
Q3_9	
Q3_10	
Q4_1	
Q4_2	
Q4_3	
Q4_4	
Q4_5	
Q4_6	
Q4_7	
Q5_1	
Q5_2	
Q6_1	
Q6_2	
Q6_3	
Q6_4	
Q6_5	
Q6_6	
Q6_7	
Q7_1	
Q7_2	
Q7_3	
Q7_4	
Q7_5	
Q7_6	
Q7_7	
Q7_8	
Q7_9	
Q7_10	
Q8_1	
Q8_2	
Q8_3	
Q9_1	
Q9_2	
Q9_3	
Q9_4	