# IS0020 Program Design and Software Tools
# Midterm, Fall, 2004

---

Name:

---

## Instruction

There are two parts in this test.

- The first part contains 22 questions worth 40 points – you need to get 20 right to get the full points. (*Estimated max time: 25 Min*)
- The second part contains 6 questions worth total of 60 points.(*Estimated max time: 40 Min*)

*Good Luck*!

---

## Part I
## (Total marks is 40)

1. The *compile* stage is when _____.
   (a) the object code is linked with code for functions in other files
   (b) the C++ program is translated into machine language code
   (c) the program is executed one instruction at a time
   (d) the program is placed in memory

2. In which of the following is `y` not equal to `5` after execution? Assume `x` is equal to `4`.
   (a) `y = 5;`
   (b) `y = x++;`
   (c) `y = ++x;`
   (d) `y = x = 5;`

3. Variables are also known as
   (a) *lvalues*, but can be used as *rvalues*
   (b) *lvalues*, and cannot be used as *rvalues*
   (c) *rvalues*, and cannot be used as *lvalues*
   (d) constant variables

4. An identifier's storage class
   (a) determines the period during which that identifier exists in memory
   (b) determines whether an identifier is known only in the current source file or in any source file with proper declarations
   (c) determines where the identifier can be referenced in a program
   (d) all of the above

5. Which of the following is not true of `static` local variables?
   (a) they are accessible outside of the function in which they are defined
   (b) they retain their values when the function is exited
   (c) they are initialized to zero if not explicitly initialized by the programmer
   (d) they can be of type `int`

6. A reference parameter
   (a) is an alias for its corresponding argument

(b) is declared by following the parameter's type in the function prototype by an ampersand (**&**)
(c) cannot be modified
(d) both (a) and (b)

7. What value does function **whatDoIOutput** return when called with a value of 4?

```
int whatDoIOutput ( int number ) {
      if ( number <= 1 )
            return 1;
      else
            return number * whatDoIOutput( number – 1 );
}
```

(a) 1
(b) 24
(c) 0
(d) 4

8. Three of the following expressions have the same value. Which of the following's value is different from the others?
(a) ***&Ptr**
(b) **&*Ptr**
(c) ***Ptr**
(d) **Ptr**

9. Assuming that **t** is an array and **tPtr** is a pointer to that array, what expression refers to the address of the fourth element?
(a) ***( tPtr + 3 )**
(b) **tPtr[ 3 ]**
(c) **&tPtr[ 3 ]**
(d) ***( t + 3 )**

10. Non-**static** member variables declared **private**
(a) can never be accessed directly or indirectly by the client.
(b) can never be modified directly or indirectly by the client.
(c) can be accessed and/or modified by any object of a different class.
(d) can be accessed and/or modified by **public** member functions and by **friend**s of the class.

11. A class may contain multiple constructors if
(a) they have different names.
(b) they have different argument lists.
(c) they have the same argument list.
(d) they have different return types.

12. Returning references to non-**const private** data
(a) allows **private** functions to be modified.
(b) is only dangerous if the binary scope resolution operator (**::**) is used in the function prototype
(c) allows **private** member variables to be modified, thus "breaking encapsulation."
(d) results in a compiler error.

13. If the line **friend class A;** appears in **class B**, and **friend class B;** appears in **class C** then
(a) **class A** is a friend of **class C**.
(b) **class C** can call **class A**'s **private** member functions.
(c) **class A** can access **private** variables of **class B**.
(d) **class B** can access **class A**'s **private** variables.

14. The code fragment

```
Increment::Increment(int c, int i) : increment (i) {
      count = c;
}
```

tells you

(a) `Increment` is a `const` variable.
(b) `Increment` must be a `const` function.
(c) `increment` may or may not be a destructor.
(d) `increment` may be a `const` variable.

15. If the functions `a()`, `b()` and `c()` all return references to an object `Test` (using the `this` pointer) and function `d()` is declared `void`, which of the following statements has correct syntax?

(a) `a().b().Test;`
(b) `Test.d().c();`
(c) `Test.a().Test.d();`
(d) `Test.a().b().d();`

16. `static` member functions:

(a) can use the `this` pointer.
(b) can only access other `static` member functions and `static` variables.
(c) cannot be called until their class is instantiated.
(d) can be declared `const` as well.

17. An overloaded `+` operator takes a class object and a `double` as operands. For it to be commutative (i.e., `a + b` and `b + a` both work),

(a) `operator+` must be a member function of the class from which the objects are instantiated.
(b) `operator+` must be a non-member function.
(c) the `operator+` function that takes the object as the left operand must be a member function, and the other `operator+` must be a non-member function.
(d) both `operator+` functions must be non-member `friend` functions of the class.

18. Suppose you have a programmer-defined data type `Data` and want to overload the `<<` operator to output your data type to the screen in the form "`cout << dataToPrint";` and allow cascaded function calls. The first line of the function definition would be

(a) `ostream operator<<(ostream &output, const Data &dataToPrint)`
(b) `ostream &operator<<(const Data &dataToPrint, ostream &output)`
(c) `ostream operator<<(const Data &dataToPrint,ostream &output)`
(d) `ostream &operator<<(ostream &output, const Data &dataToPrint)`

19. There exists a data type `Date` and member function `Increment`. The `++` operator is being overloaded to postincrement an object of type `Date`. Select the correct implementation.

(a)
```
Date Date::operator++( int ) {
      Date temp = *this;
      Increment();
      return *temp;
}
```
(b)
```
Date Date::operator++( int ) {
      Increment();
      Date temp = *this;
      return temp;
}
```
(c)
```
Date Date::operator++( int ) {
      Date temp = *this;
      return this;
```

```
        temp.Increment();
}
```
(d)
```
Date Date::operator++( int ) {
        Date temp = *this;
        Increment();
        return temp;
}
```

20. Inside a function definition, and for an object with data element **x**, which of the following is not equivalent to **this->x** ?
    (a) ***this.x**
    (b) **(*this).x**
    (c) **x**
    (d) **(* (& (*this) ) ).x**

21. The **delete** operator
    (a) can terminate the program.
    (b) can delete an entire array of objects declared using **new**.
    (c) must be told which destructor to call when destroying an object.
    (d) is called implicitly at the end of a program.

22. **static** member functions:
    (a) can use the **this** pointer.
    (b) can only access other **static** member functions and **static** variables.
    (c) cannot be called until their class is instantiated.
    (d) can be declared **const** as well.

## Part II
## Total points 10 x 6 = 60

1. Write the output for the following program

```
int i;
for ( i= 2; i < 10; --i){
      cout << "Here:  " << i << endl;
      i = i + 2;
}
return 0;
```

**Output**:

Here: 1
Here: 2
Here: 9

2. Write the output for the following program module

```
int sum = 0;
int i = 1, j = 1;
while ( (i = 5) != 10) {
      sum = sum + i;
      if (sum == 20) break;
      j++;
}
cout << i << ":" << j << endl;
```

**Output**: 5.4

3. Give the output of the following program module.

```
int main() {
      int count = 10;
      int &cRef = count;
      int i = 0;
      while (count < 1000) {
            if (i) {
                  cRef = cRef * 10;
                  i = 0;
            }
            else {
                  count = count * 10;
                  i = 1;
            }
            cout << (++count)++ << endln;
      }
      cout << (++count)++ << endln;
      return 0;
}
```

**Output**:

```
101
1021
1023
```

4. What is the output of the following program:

```
int x = 500;
main(){
      int x = 100;
      { char x = 'Z';
            {     int x = 300;
                  cout << "One: " << x << " " << ::x << endl;
            }
            cout << "Two: " << x << " " << ::x << endl;
      }
      return 0;
}
```

**Output:**

```
One: 300 500
Two: Z  500
```

1. Consider class `CreateAndDestroy` that we discussed in class – it has a *constructor* and a *destructor* function. When an object of this class is created its *constructor* takes two strings as arguments and prints a message. For example, the statement

```
CreateAndDestroy c(1, “(In Example)”);
```

would call its *constructor*, which will simply print out the message

```
Object 1 constructor runs <In Example>
```

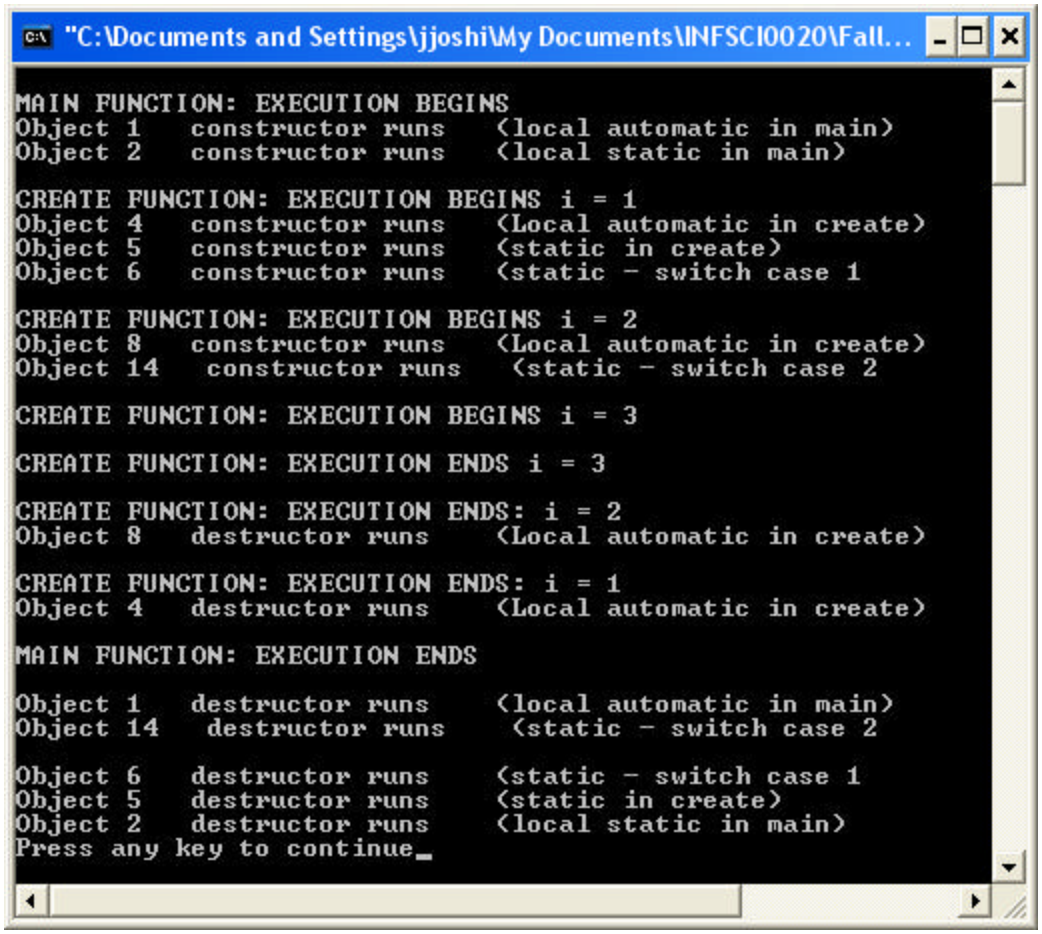
Its *destructor* will, on the other hand will simply print out the message

```
Object 1 destructor runs <In Example>
```

Write the sequence of creation and destruction of objects if the following code is run. ***Write exactly what it would print***.

```
int main()
{
   cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy first( 1, "(local automatic in main)" );
   static CreateAndDestroy second(2, "(local static in main)" );
   create(1);
   cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
   return 0;
} // end main

// function to create objects
void create(int i)
{
   cout << "\nCREATE FUNCTION: EXECUTION BEGINS i = " << i << endl;
   if (i > 2) {
        cout << "\nCREATE FUNCTION: EXECUTION ENDS i = " << i << endl;
        return;
   }
   CreateAndDestroy x1( i*4, "(Local automatic in create)" );
   static CreateAndDestroy fifth( i*5, "(static in create)" );
   switch(i) {
       case 1:
       {     static CreateAndDestroy x2(i*6, "(static - switch case 1");
             create(i+1);
             break;
       }
       case 2:
       {     static CreateAndDestroy x3(i*7, "(static - switch case 2");
             create(i+1);
             break;
       }
       default:
             static CreateAndDestroy x4(++i*8, "(static - DEFAULT");
             create(i++);
             break;
   }
   cout << "\nCREATE FUNCTION: EXECUTION ENDS: i = " << i << endl;
} // end function create
```




2. Consider the following two statement with regards to your `Complex` class implementation

   A. `Complex (*XXptr[5])(Complex &) = {add, substract, multiply}`

   B. `Complex (*XXptr[5])(Complex &) = {add, substract, multiply, equal}`

   Would they be considered syntactically correct statements? If they are correct, describe what the statements mean? If not, state what is the wrong with the statements ?