Below, separated by %%%%% signs, you will find 3 files. The first is a java source file for CPoolTest.java, a simple java program that exemplifies many of the concepts we discuss in the previous classes. I refer to this below as the java source file (which I compile via a source compiler to CPoolTest.class).

The second is the output of a program called ShowMeTheStructure (SMTS) that the instructor wrote using the BIT java classfile parser that extracts the information about a bytecode classfile. The input used to produce this output is CPoolTest.class. I refer to this output below as the "constant pool dump". The SMTS tool and build instructions can be found on the course schedule page.

Third is the list of bytecode instructions in each method CPoolTest.class. We refer to this as the "bytecode instructions list".

ASSIGNMENT
In the bytecode instructions list below, you will see bytecode instructions in each method (don't worry if you don't understand them -- we will learn about them in a future class). Some instructions access the constant pool. When they do so they use the hash sign to identify the index into the constant pool. Your job is to specify (using a specific format) what each constant pool index refers to for each instruction listed below.

The format is "type value". Valid entries for "type" are: string, float, long, double, integer, method, field, class. Specify type using lowercase characters. The format for value is either the actual value (for all but field and method types) or class_name.member_name:member_type for field and method members/types. Use the exact values (and case) for class_name, member_name, and member_type that are specified in the constant pool dump. Similarly for value, specify the exact value as given in the constant pool dump.

For example in static method main:
0: ldc           #6
is an instruction (with byte index 0 in main, called the load_constant bytecode instruction) that accesses constant pool index 6 (denoted by #6). We lookup this index in the constant pool dump (each index starts at a line prefixed with "Index:" in this section). Some index types have additional indexes encoded in the entry -- which you must chase to get all of the information for the answer. Other indexes are "leaf" indexes and simply store a literal value (UTF8 string, integer, float, double, long). In this example, the answer is "float 3.24f". We find it by looking up index 6 in the constant pool dump, seeing that its type is Float (and there are no additional indexes to chase) and noting its value at that index (3.24f).

Another example is in instance method testme():
5: invokevirtual #5
This is an instruction at byte index 5 in testme that invokes a virtual (dynamically dispatched) method via the invokevirtual bytecode instruction. This instruction accesses index 5 in the constant pool (denoted by #5). In this case, the answer is "method java/io/PrintStream.println:(Ljava/lang/String;)V".

To get this answer, we lookup index 5 in the constant pool dump and see that its entry has type CONSTANT_MethodRef (which tells us that the type is "method", which points to two additional indexes (31 of type CONSTANT_Class and 32 of type CONSTANT_NameAndType). We chase 31 and find that it holds index 40 of type CONSTANT_Utf8. We chase this index and find that it does not hold any more indexes (is a leaf index) and instead holds a string value "java/io/PrintStream", which is the class_name of the class that defines the method being called (because we started chasing 31 of type CONSTANT_class).

We next chase index 32 to find that it ultimately refers to one leaf index for the Name of the method being invoked, with the string "println". The second leaf index at this entry (index 32) refers to another leaf index for the Type of the method being invoked by this instruction. It points to the string "(Ljava/lang/String;)V". "println" is the method_name and "(Ljava/lang/String;)V" is the method_type (ie its signature). We put all of this together to form: type class_name.method_name:method_type, which in this case is our answer: method java/io/PrintStream.println:(Ljava/lang/String;)V

Instructions:
Your answer must use this form: type value
Valid entries for "type" are: string, float, long, double, integer, method, field, class (use all-lowercase). Construct the value either directly from the constant pool dump or by chasing the indexes as described above. If the type is method, specify the value as class_name.method_name:method_type (using values for each of these terms as specified exactly by the constant pool dump). If the type is field, specify the value as class_name.field_name:field_type (using values for each of these terms as specified exactly by the constant pool dump). For all other type, specify the value given in the constant pool dump directly at the leaf index.

java source file: CPoolTest.java

```java
class CPoolTest {
    /* Instance fields */
    int field1 = 2;

    /* Instance methods */
    void testme() {
        System.err.println("test:testme");
    }
}
```
/* Static fields */
    static int sfield1; /* fields without assignments, get default values (here=0) */

/* Static/class methods */
    public static void main(String a[]) {
        float val = 3.24f; /* variable of type float holds a float literal */
        double val2 = 100000.246d; /* variable of type float holds a double literal */
        CPoolTest.sfield1 = 17; /* variable of type float holds a double literal */

        /* allocate & instantiate object of type CPoolTest, store reference in variable obj */
        CPoolTest obj = new CPoolTest();

        int i = obj.field1; /* access instance field1 in object */
        obj.testme(); /* invoke instance method testme on object */
    }

The SMTS tool can be found here: https://sites.cs.ucsb.edu/~ckrintz/classes/s20/cs263/showme.tar.gz
The steps for building/using can be found here: https://sites.cs.ucsb.edu/~ckrintz/classes/s20/cs263/showme-steps.txt

Output (with CPoolTest.class as input)

class: /Users/ckrintz/src/java/cs263/showme/cs263/../../CPoolTest.class
    Class uses the current invokspecial semantics
    Class is package access
    Interfaces implemented: (none if 0) 0
    Attributes implemented: (none if 0) 1
    SourceFile Attribute
        uses index: 24 (name), and 25 (source file name)

Constant Pool: *****************************************
Index: 0 is not used and must be set to null (JVM Specification)
Index: 1
    CONSTANT MethodRef: 1
        Points to CONSTANT Class index: 13
        AND Points to CONSTANT NameAndType index: 26
Index: 2
    CONSTANT FieldRef:
        Points to CONSTANT Class index: 10
AND Points to CONSTANT_NameAndType index: 27

Index: 3
CONSTANT_FieldRef:
  Points to CONSTANT_Class index: 28
  AND Points to CONSTANT_NameAndType index: 29

Index: 4
String:
  CONSTANT_Utf8 index: 30

Index: 5
CONSTANT_MethodRef: 5
  Points to CONSTANT_Class index: 31
  AND Points to CONSTANT_NameAndType index: 32

Index: 6
Float: value: 3.24f

Index: 7
Double:
  Index: 7 and 8 are cpool entries for double
  Double: value: 100000.246d

Index: 8
NullInfo:

Index: 9
CONSTANT_FieldRef:
  Points to CONSTANT_Class index: 10
  AND Points to CONSTANT_NameAndType index: 33

Index: 10
CONSTANT_Class:
  CONSTANT_Utf8 index: 34

Index: 11
CONSTANT_MethodRef: 11
  Points to CONSTANT_Class index: 10
  AND Points to CONSTANT_NameAndType index: 26

Index: 12
CONSTANT_MethodRef: 12
  Points to CONSTANT_Class index: 10
  AND Points to CONSTANT_NameAndType index: 35

Index: 13
CONSTANT_Class:
  CONSTANT_Utf8 index: 36

Index: 14
CONSTANT_Utf8:  string value: field1

Index: 15
CONSTANT_Utf8:  string value: I

Index: 16
CONSTANT_Utf8:  string value: sfield1

Index: 17
CONSTANT_Utf8:  string value: <init>

Index: 18
CONSTANT_Utf8:  string value: ()V

Index: 19
CONSTANT_Utf8:  string value: Code
CONSTANT_Utf8: string value: LineNumberTable
CONSTANT_Utf8: string value: testme
CONSTANT_Utf8: string value: main
CONSTANT_Utf8: string value: ([Ljava/lang/String;)V
CONSTANT_Utf8: string value: SourceFile
CONSTANT_Utf8: string value: CPoolTest.java
CONSTANT_NameAndType: 26
  CONSTANT_Utf8 name index: 17
  AND CONSTANT_Utf8 descriptor/type index: 18
CONSTANT_NameAndType: 27
  CONSTANT_Utf8 name index: 14
  AND CONSTANT_Utf8 descriptor/type index: 15
CONSTANT_NameAndType: 29
  CONSTANT_Utf8 name index: 38
  AND CONSTANT_Utf8 descriptor/type index: 39
CONSTANT_Utf8: string value: test:testme
CONSTANT_NameAndType: 33
  CONSTANT_Utf8 name index: 16
  AND CONSTANT_Utf8 descriptor/type index: 15
CONSTANT_Utf8: string value: CPoolTest
CONSTANT_NameAndType: 35
  CONSTANT_Utf8 name index: 21
  AND CONSTANT_Utf8 descriptor/type index: 18
CONSTANT_Utf8: string value: java/lang/Object
CONSTANT_Utf8: string value: java/lang/System
Looping thru the fields this class defines (if any)...

Field defined in this class, name index: 14 descriptor index: 15
access flags 0

Field defined in this class, name index: 16 descriptor index: 15
access flags 8 static

Looping thru the methods this class defines (if any)...

Method defined in this class, name index: 17 descriptor index: 18
Attribute list:
Attrib name index: 19
    Its a Code Attribute and it has 1 attributes
    index: 20
    Exception Table for this code attribute has length: 0 (number
of try blocks)

Method defined in this class, name index: 21 descriptor index: 18
Attribute list:
Attrib name index: 19
    Its a Code Attribute and it has 1 attributes
    index: 20
    Exception Table for this code attribute has length: 0 (number
of try blocks)
Method defined in this class, name index: 22 descriptor index: 23
static
Attribute list:
Attrib name index: 19
   Its a Code Attribute and it has 1 attributes
      index: 20
   Exception Table for this code attribute has length: 0 (number
of try blocks)
******************************

Summary:
classfile: /Users/ckrintz/src/java/cs263/showme/cs263/../../
CPoolTest.class
code length in bytes: 53
symbolic data size in bytes: 450
total size of classfile: 625
total size of All classfiles so far: 625

%%%%%%%% bytecode instructions %%%%%%%%%%%
class CPoolTest {

    CPoolTest();
    Code:
       0: aload_0
       1: invokespecial #1
       4: aload_0
       5: iconst_2
       6: putfield      #2
       9: return

    void testme();
    Code:
       0: getstatic     #3
       3: ldc           #4
       5: invokevirtual #5
       8: return

    public static void main(java.lang.String[]);
    Code:
       0: ldc           #6
       2: fstore_1
       3: ldc2_w        #7
       6: dstore_2
       7: bipush 17
       9: putstatic     #9
      12: new           #10
      15: dup
      16: invokespecial #11
      19: astore 4


21: aload 4
23: getfield       #2
26: istore 5
28: aload 4
30: invokevirtual #12
33: return