

CSE 222 — Midterm Exam

SAMPLE

This is a closed-book, closed-notes, closed-neighbor exam.

In accordance with The Ohio State University Code of Student Conduct, I certify that I have neither received nor given aid on this examination, that I shall not discuss the contents of this examination with anyone in CSE 222 who has not already taken the exam, and that I have not written down and taken from the room any questions or answers from this exam.

Name _____

Signature _____

There are 100 points on the exam. Please write your answers on the test sheets, and of course don't forget to write **and sign** your name on the top sheet at least. Consider the space allotted as an indication of the expected length of the answer.

1. (12 points; 3 points each) In front of each problem description, write the letter of the software component family you could *best* use to address it, in the sense that it would call for the least client programming. Each letter should be used exactly once.

A. Partial_Map

C. Record

B. Queue

D. Set

___ You need to keep together several pieces of information about a computer system order: manufacturer name, processor speed, amount of memory, amount of disk space, and price.

___ You need to keep track of all the ZIP codes in the Eastern time zone in a program used by a Columbus mail-order company, so the program can tell whether a particular ZIP code is on Eastern time.

___ You need to keep track of hotel guests and their hotel-room phone numbers, so the hotel operator can ask a caller for a guest's name and input that information to connect the call to the guest's room.

___ You need to simulate the service times in a fast-food restaurant by, among other things, keeping track of the customers waiting in line to be served.

2. (16 points; 4 points each)

(a) Suppose $s1$ and $s2$ are finite mathematical sets of integers, and consider this assertion:

```
there exists x, y: integer, a: finite set of integer  
  (s1 = {x} union a and  
   s2 = {y} union a)
```

Which one of the following pairs of values for $s1$ and $s2$ makes the assertion true?

- A. $s1 = \{5, 6, 7\}$, $s2 = \{5, 6, 8\}$
 - B. $s1 = \{5, 6, 7\}$, $s2 = \{5, 8, 9\}$
 - C. $s1 = \{1, 2, 8, 9\}$, $s2 = \{1, 2, 3\}$
 - D. $s1 = \{4, 5, 6\}$, $s2 = \{4, 5, 6, 7, 8\}$
- (b) Suppose you need to sort into alphabetical order the elements currently being kept in an object of type `Set_of_Text` (instantiated as `Set_Kernel_1a_C <Text>`). Assume that an appropriately named utility class containing `Are_In_Order` has already been created. Which concrete instance would be useful? Circle the best answer.

A.

```
concrete_instance  
class Sorter:  
  instantiates Sorting_Machine_Kernel_1a_C <  
    Set_of_Text,  
    Text  
  >  
  
  {};
```

B.

```
concrete_instance  
class Sorter_of_Set_of_Text:  
  instantiates Sorting_Machine_Kernel_1a_C <  
    Set_of_Text,  
    Set_of_Text_Are_In_Order_1  
  >  
  
  {};
```

C.

```
concrete_instance  
class Sorter_of_Text:  
  instantiates Sorting_Machine_Kernel_1a_C <  
    Text,  
    Text_Are_In_Order_1  
  >  
  
  {};
```

D. none of the above

- (c) How many calls to Foo (including the first one, *Foo(5)*) actually result from the call *Foo(5)*? Stated another way, how many tracing tables would you need to trace the call *Foo(5)* if you traced into the body of *Foo* on each call? **To the right of the code, draw a picture that shows how you got your answer.** Here is the body:

```

function_body Integer Foo (
    preserves Integer n
)
{
    if (n <= 2)
    {
        return n;
    }
    else
    {
        return Foo (n-2) + Foo (n-3);
    }
}

```

- (d) A colleague has asked you to review some code in the concrete template class *Sequence_Copy_1 <Item>* for duplicating a sequence. (Only the relevant parts, i.e., the procedure specification and the procedure body code between {...}, are shown.) Discuss **all** the errors you find in the procedure body, or give a short statement explaining why you think it is correct.

```

procedure Copy_To (produces Sequence_Copy_1& copy);
/*!
    preserves self
    ensures
        copy = self
!*/

```

The procedure body:

```

{
    object Integer i;
    while (i < self.Length())
    {
        copy.Add (i, self[i]);
        i++;
    }
}

```

3. (27 points) There are several parts to this question. They refer to a single situation, but you should be able to answer each subquestion independently of the others. Consider the following operation in the public interface of abstract template class *Queue_Weird*, which extends *Queue_Kernel*:

```

procedure Weird (
    produces Item& y
) is_abstract;
/*!
    requires
        |self| >= 2
    ensures
        there exists x: Item, a: string of Item
        (#self = <x> * <y> * a and
         self = a * <x>)
!*/

```

- (a) (3 points) Suppose (for parts (a) and (b) of the question *only*) that *Item* is replaced by *Integer*. Which of the following pairs of values for *self* and *y* are valid inputs to *Weird*? Circle the valid inputs; cross out the invalid inputs.

self = < > y = -2	self = <3, 2> y = 1	self = <6, 3, 4, 5, 6, 7, 8> y = 6
----------------------	------------------------	---------------------------------------

- (b) (4 points) Complete the tracing table:

Statement	Object Values
	q = <99, -6, 13, 59> n = 4
q.Weird (n);	

- (c) (6 points) Draw the CCD that depicts the direct coupling between *Queue_Weird_13* (code shown on the next page) and the other components it depends on.

- (d) (3 points) Normally, the “black box” that you (should) have drawn in part (c) is not labelled with a name in the CCD. If it were labelled, what name should it have in order to be consistent with the names used in the code for *Queue_Weird_13*?
- (e) (3 points) Why is the black box shaped like a rectangle, not a round-cornered rectangle?
- (f) (8 points) Write an appropriate procedure body to implement *Weird* that could go in *Queue_Weird_13*.

```

concrete_template <
    concrete_instance class Item,
    concrete_instance class Queue_Base
    /*!
        implements
            abstract_instance Queue_Kernel <Item>
    !*/
    >
class Queue_Weird_13 :
    implements
        abstract_instance Queue_Weird <Item>,
    extends
        concrete_instance Queue_Base
    {
public:

        procedure_body Weird (
            produces Item& y
        )
        {

        }
    }
};

```

4. (45 points) There are several parts to this question. They refer to a single situation, and each part assumes the existence of the declarations that precede it. Your firm has been hired by an art museum. Among the many tasks to be completed is this one: given a file containing information about all the works of art the museum currently displays, read from *stdin* a list of the ID numbers of works that are to be taken off display; then write to *stdout* a report showing the works that are to be taken off display, sorted by ID number.

(a) (3 points) Suppose the report generator program that you are about to write is in the file “Midterm.cpp”, and that you compile it and create an executable using the “Build Project” command. Write a Unix command that runs the resulting executable program, redirecting *stdin* from the file `unpopular.txt`.

(b) (3 points) Complete the declaration needed to instantiate a component that defines a type called *Art_Record*, which consists of a *Text* object called “title”, a *Text* object called “artist”, and an *Integer* object called “year”.

```

concrete_instance
class Art_Record :
    instantiates Record <
        _____,
        _____,
        _____
    >
{};

```

(c) (2 points) Write the statement that declares the field name for the “title” field of *Art_Record*.

```

field_name ( _____, _____, _____, _____ );

```

(d) (4 points) Complete the declaration needed to instantiate a component that defines a type called *Art_Map*, which maps each work’s unique *Integer* ID to the associated *Art_Record*.

```

concrete_instance
class Art_Map:
    instantiates Partial_Map_Kernel_1_C <
        _____,
        _____
    >
{};

```

- (e) (4 points) Consider the following object declarations:

```
object Art_Map now_on_display;  
object Integer id;
```

What are the types, respectively, of the following two objects?

- `now_on_display[id]`
- `now_on_display[id][artist][0]`

- (f) (4 points) The following declaration creates a concrete instance that will allow you to sort the *Integer* IDs into increasing order for the report.

```
concrete_instance  
class Sorter_Of_Art_Work_IDs :  
    instantiates Sorting_Machine_Kernel_1_C <  
        Integer,  
        Integer_Are_In_Order_1  
    >  
{};
```

In the above template instantiation, two actual parameters are passed to *Sorting_Machine_Kernel_1_C*. Briefly explain the purpose of each parameter. That is, what do these two parameters mean to *Sorting_Machine_Kernel_1_C*?

- (g) (25 points) Complete the body of the main program on page 9. You may assume that the program begins with all of the declarations discussed in previous parts of this problem. Details about the input and desired output are given below.

The program must do three things:

- read from a file called “art.txt” the list of all art works the museum currently has on display;
- read from *stdin* the list of works (from among those above) that are to be taken off display;
- write to *stdout* a report showing all the works to be taken off display—**listed in increasing order by ID**—along with their associated titles and artists.

The format of the “art.txt” file is one field per line. In particular, the file contains one piece of information per line for each work of art: a unique ID, title, artist, and year. For example, the contents of “art.txt” might look like:

```
913960
Pond of Crustaceans
Clawed Monet
1872
432554
Behind the Bushes
Dick Halliburton
2000
680713
Di Van Japonais
Tooloose Totrack
1894
632565
Musical Instrunuts
Pablo Pistachio
1912
```

The input from *stdin* is simpler. It just contains one ID per line for each work of art to be taken off display, in no particular order. Assume that all the IDs read from *stdin* are among those read from “art.txt”. This input might look like:

```
680713
432554
```

Your program must generate a report that looks like this (for the sample input above):

```
WORKS TO BE TAKEN OFF DISPLAY
432554: Behind the Bushes (Dick Halliburton)
680713: Di Van Japonais (Tooloose Totrack)
```

*The area below is scratch space.
Complete the program on page 9.*

```
// Assume you have here all the appropriate #includes and the
// instantiations from all previous parts of this problem.

procedure_body main ()
{
    object Character_IStream artfile, input;
    object Character_OStream output;
    object Art_Map now_on_display;
    object Sorter_Of_Art_Work_IDs sorter;

    artfile.Open_External ("art.txt");      // to read from "art.txt"
    input.Open_External ("");              // to read from stdin
    output.Open_External ("");             // to write to stdout
```

```
    artfile.Close_External ();
    input.Close_External ();
    output.Close_External ();
}
```

*This page is scratch space.
On the next sheet is a Component Operation Summary.
Please do not detach it from the rest of the exam.*

CSE 222 Component Operation Summary

Please return this sheet with your exam. If you need to see the interface contract specification for any component during the exam, please ask your instructor.

Partial_Map_Kernel

Define (**consumes** D_Item& d, **consumes** R_Item& r)
Undefine (**preserves** D_Item& d, **produces** D_Item& d_copy, **produces** R_Item& r)
Undefine_Any (**produces** D_Item& d, **produces** R_Item& r)
[**preserves** D_Item& d] (names the R_Item associated with d)
Boolean Is_Defined (**preserves** D_Item& d)
Integer Size ()

Queue_Kernel

Enqueue (**consumes** Item& x)
Dequeue (**produces** Item& x)
[**current**] (names the Item at the front, or left, end)
Integer Length ()

Sequence_Kernel

Add (**preserves** Integer pos, **consumes** Item& x)
Remove (**preserves** Integer pos, **produces** Item& x)
[**preserves** Integer pos] (names the Item at position pos)
Integer Length ()

Set_Kernel

Add (**consumes** Item& x)
Remove (**preserves** Item& x, **produces** Item& x_copy)
Remove_Any (**produces** Item& x)
Boolean Is_Member (**preserves** Item& x)
Integer Size ()

Sorting_Machine_Kernel

Insert (**consumes** Item& x)
Change_To_Extraction_Phase ()
Remove_First (**produces** Item& x)
Remove_Any (**produces** Item& x)
Boolean Is_In_Extraction_Phase ()
Integer Size ()

Stack_Kernel

Push (**consumes** Item& x)
Pop (**produces** Item& x)
[**current**] (names the Item at the top, or left, end)
Integer Length ()