1.

a.

	alicerc	*bobrc*	*cyndyrc*
Alice	ox	r	
Bob	r	ox	
Cyndy	r	rw	orwx

a.

	alicerc	*bobrc*	*cyndyrc*
Alice	ox	r	r
Bob		ox	
Cyndy	r	rw	orwx

ex2

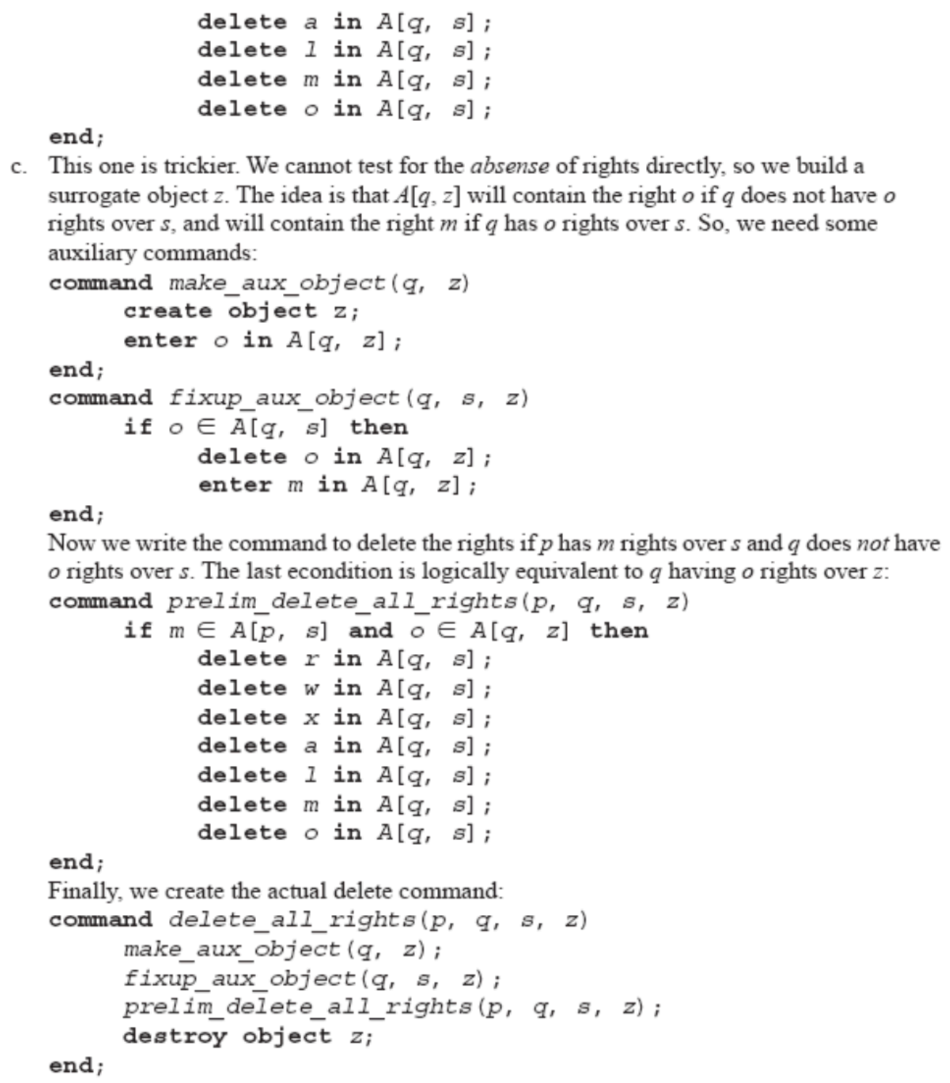
2. In these answers, *r*, *w*, *x*, *a*, *l*, *m*, and *o* represent the read, write, execute, append, list, modify, and own rights, respectively.

   a. The key observation is that anyone can delete the rights, not *p*. So:

```
command delete_all_rights(p, q, s)
          delete r in A[q, s];
          delete w in A[q, s];
          delete x in A[q, s];
          delete a in A[q, s];
          delete l in A[q, s];
          delete m in A[q, s];
          delete o in A[q, s];
end;
```

   b. Here, we must condition the command on the presence of rights that *p* has over *s*:

```
command delete_all_rights(p, q, s)
          if m ∈ A[p, s] then
                    delete r in A[q, s];
                    delete w in A[q, s];
                    delete x in A[q, s];
```



```
            delete a in A[q, s];
            delete l in A[q, s];
            delete m in A[q, s];
            delete o in A[q, s];
end;
```

c. This one is trickier. We cannot test for the *absense* of rights directly, so we build a surrogate object *z*. The idea is that *A*[*q*, *z*] will contain the right *o* if *q* does not have *o* rights over *s*, and will contain the right *m* if *q* has *o* rights over *s*. So, we need some auxiliary commands:

```
command make_aux_object(q, z)
      create object z;
      enter o in A[q, z];
end;
command fixup_aux_object(q, s, z)
      if o ∈ A[q, s] then
            delete o in A[q, z];
            enter m in A[q, z];
end;
```

Now we write the command to delete the rights if *p* has *m* rights over *s* and *q* does *not* have *o* rights over *s*. The last econdition is logically equivalent to *q* having *o* rights over *z*:

```
command prelim_delete_all_rights(p, q, s, z)
      if m ∈ A[p, s] and o ∈ A[q, z] then
            delete r in A[q, s];
            delete w in A[q, s];
            delete x in A[q, s];
            delete a in A[q, s];
            delete l in A[q, s];
            delete m in A[q, s];
            delete o in A[q, s];
end;
```

Finally, we create the actual delete command:

```
command delete_all_rights(p, q, s, z)
      make_aux_object(q, z);
      fixup_aux_object(q, s, z);
      prelim_delete_all_rights(p, q, s, z);
      destroy object z;
end;
```

- **c. Modify your command so that the deletion can occur only if p has modify rights over s and q does not have own rights over s.**

- command delete_all_rights(p,q,s)
  create subject tmp
  enter read in a[tmp,s]
  if own in a[q,s] then
  delete read from a[tmp,s]
  if modify in a[p,s] and read in a[tmp,s] then
  delete read in a[q,s]
  delete write in a[q,s]
  delete execute in a[q,s]
  delete append in a[q,s]
  delete list in a[q,s]
  delete modify in a[q,s]
  delete own in a[q,s]
  destroy subject tmp
  end

ex3

The query-set-overlap mechanism requires a history of all queries to the database. Discuss the feasibility of this control. In particular:

a) How will the size of the history affect the response of the mechanism?

Answer:

The history tracker keeps the attributes answered in the past queries. If the history is small, then the mechanism will have less possible queries and the query-set-overlap mechanism will work. However, if the history is big there is high processing overhead as every new query will need to be compared with all previous ones causing the system to slow down. In addition, the profile of all the users are required to be stored and this increases the use of storage space.

b) How practical is a system that enforces this mechanism?

Answer:

It is not practical for a system to enforce this especially in large datasets with many users as the history tracker becomes very big. The processing overhead will be high causing the system to slow down and large storage space is required. In addition, this mechanism is not effective against several users who may collude to make independent queries and later collate them to form useful information.

However, we read an interesting journal article which a particular team of researchers have developed a model to mitigate these shortcomings. On improving performance, one possible solution is to specify policies such as discard stored information whenever the database is updated. On preventing a colluded attack, one possible solution is to assume all queries from all users as events of a single infinite-length execution[3]. As we have limited time to research sufficient materials, we would conclude (based on current information) that the query-set-overlap mechanism has limitations and will not be practical for large datasets and organizational use.

ex4

ex5

Let c be a copy flag and let a computer system have the same rights as in Exercise 5 of the Classwork. The set of rights {read, write, execute, append, list, modify, own}.

a) Using the syntax in Section 2.3 of Bishop's, write a command copy_all_rights(p,q,s) that copies all rights that p has over s to q.

Answer:

```
command create_file(p,s)
    create object s;
    enter c into a[p,s];
    enter own into a[p,s];
end
command copy_all_rights(p,q,s)
     enter r into a[q,s];
     enter w into a[q,s];
     enter e into a[q,s];
     enter a into a[q,s];
     enter l into a[q,s];
     enter m into a[q,s];
     enter own into a[q,s];
end
```

You need to check if the right is available in p before u copy over to q.

b) Modify your command so that only those rights with an associated copy flag are copied. The new copy should not have the copy flag.

Answer:

```
command copy_all_rights(p,q,s)
    if c in a[p,s]
    then
     enter r into a[q,s];
     enter w into a[q,s];
     enter e into a[q,s];
     enter a into a[q,s];
     enter l into a[q,s];
     enter m into a[q,s];
end
```

You need to check if the right is available in p before u copy over to q.

The own right is not copied to the q and therefore process q does not have the rights to copy flag.

c) In part b), what conceptually would be the effect of copying the copy flag along with the right?
Answer:

If the copy flag is copied with the right, the process q is able to assign all the rights over object s to other processes.

ex6
homework-6