

## Problem Set 9

Due before midnight on Tuesday, April 24, 2007.

### 1 Rolling Admissions

A university has a policy of rolling admissions. Students apply, receive offers of admission, and either accept or decline the offers. At any given time, there is a pool of students who have received offers of admission but have not yet accepted or declined the offers.

This university has a limited number of merit-based scholarships which it uses to attract and retain top students. Students are eligible for the scholarship only if they have been admitted, have not already been offered a scholarship, and have a combined SAT score of at least 1100. Scholarships are awarded on a rolling basis. Whenever a scholarship becomes available, it is offered to the eligible student in the pool with the highest SAT score. Ties are broken in favor of the student who was admitted first. If there are no eligible students, any unassigned scholarships are held in abeyance until additional eligible students are accepted.

### 2 Action File

Admissions data is furnished to the program in the form of an actions file, each line of which describes an admissions action or event. The four kinds of actions are:

**admit** Adds a student to the poll of admitted students. If eligible for a scholarship, the student is also put on the scholarship queue.

An admit line comprises four whitespace-separated fields:

1. The word “admit”.
2. The student’s name, enclosed in double quotes.
3. A unique student ID consisting of a string of non-whitespace characters.
4. The student’s SAT score.

It is an error if the ID collides with the ID of another student in the pool.

Example:

```
admit "Kevin Massey"      2007F0003 1423
```

**accept** Marks that the student has accepted the offer of admission.

An accept line comprises two whitespace-separated fields:

1. The word “accept”.
2. A student ID number.

It is an error if the ID does not match any student in the pool.

Example:

```
accept 2007F0003
```

**decline** Marks that the student has declined the offer of admission. If the student had been offered a scholarship, the scholarship is released and made available for reassignment. The student is removed from the scholarship queue, if present, and removed from the pool of admitted students.

A decline line comprises two whitespace-separated fields:

1. The word “decline”.
2. A student ID.

It is an error if the ID does not match any student in the pool.

Example:

```
decline 2007F0003
```

**available** Adds to the count of currently available scholarships.

An available line comprises two whitespace-separated fields:

1. The word “available”.
2. An integer  $n$  giving the number of additional scholarships to be made available.

Example:

```
available 5
```

### 3 Required Output

The goal of this assignment and the next is to write a program that manages and assigns scholarships to students according to the rules described above. The program should print a line of output whenever the following events occur:

- A student is offered a scholarship.
- Scholarship student accepts the offer of admission.
- Scholarship student declines the offer of admission.
- New scholarships become available.

In the first three cases, the line printed should begin with the words “Scholarship *event*” followed by the student’s name, ID number, and SAT score, where *event* is replaced by “offered to”, “accepted by”, or “declined by” as appropriate. For example, when John Smith is offered a scholarship, the following would be printed:

```
Scholarship offered to John Smith (ID=2007F0001, SAT=1350)
```

When new scholarships become available, a line should be printed showing the number of new scholarships and the total number of unfilled scholarships, including the new ones. For example, if there were two unfilled scholarships and a student with a scholarship declined, the following line would be printed:

```
1 new scholarships, 3 unfilled
```

Note that a scholarship released by a student who declines is considered to be “new”.

When end of file is reached, the program should print out the following data:

- The list of students currently in the applicant pool.
- Number of scholarships currently unassigned.
- Number of scholarships offered to students who have not yet responded to the offer.
- Number of scholarships offered and accepted.
- Number of scholarships offered and declined.
- The list of eligible students who have not received a scholarship.

## 4 Assignment

Using the data structures from problem set 8, write a command `action` that reads admissions action data from a file, awards scholarships as described in sections 1–2, and produces output as described in section 3.

## 5 Detailed specification

Your program as usual should be built from several modules.

**Module `driver.c`** contains the main program which performs command-line processing, opens and closes the input file, creates an `action` object, and call an action function to process the input file.

**Module `action`** reads and parses the action file and identifies and handles the events. It uses modules `pool` and `scholarship` to carry out its actions.

**Module `pool`** maintains the pool of admitted students. It should implement the pool using the `syntab` module from problem set 8. If done properly, no changes should be necessary to `syntab`; `pool` will just act as an interface.

**Module `scholarship`** manages the count of available scholarships and assigns them to eligible students. It has functions for awarding scholarships and producing the scholarship activity printout. It should have a function `award_scholarship()` that is called by the action module after each event is processed. Whenever it is called, `award_scholarship()` assigns scholarships to eligible students until either all available scholarships have been assigned or all eligible students have been awarded a scholarship. It uses module `eligible` to manage the list of students awaiting scholarships.

**Module `eligible`** maintains the priority queue of eligible students waiting for scholarships. It should implement the queue using the `heap` module from problem set 8. If done properly, no changes should be necessary to `heap`; `eligible` will just act as an interface.

**Module `student`** manages a student record. It defines type `student` to point to a struct containing the student's name, ID number, SAT score, and other relevant information for carrying out this assignment.

You are encouraged to make use of code previously written in this course. For example, `flexbuf` of problem set 4 is useful for reading lines from the admissions data file.

I will place sample input and output files in in the course directory

```
/c/cs223/assignments/ps9
```

They will be useful for the initial testing of your program, but they are by no means complete. In particular, you should include tests that have unassigned scholarships at the end, tests that have eligible students waiting for scholarships, tests that show the scholarships are indeed being offered to the highest priority students when there is a choice, and so forth.

## 6 Deliverables

You should submit source code and a `Makefile` for all of your modules (including `syntab` and `heap`) along with test input and output files which test your code as well as possible.