An operating system's **pid manager** is responsible for managing process identifiers. When a process is first created, it is assigned a unique pid by the pid manager. The pid is returned to the pid manager when the process completes execution, and the manager may later reassign this pid. Process identifiers must be unique; no two active processes may have the same pid.

Use the following constants to identify the range of possible pid values:

```
#define MIN_PID 300
#define MAX_PID 5000
```

You may use any data structure of your choice to represent the availability of process identifiers. One strategy is to adopt what Linux has done and use a bitmap in which a value of 0 at position *i* indicates that a process id of value *i* is available and a value of 1 indicates that the process id is currently in use.

Implement the following API for obtaining and releasing a pid:

- int allocate_map(void) – Creates and initializes a data structure for representing pids; returns -1 if unsuccessful and 1 if successful
- int allocate_pid(void) – Allocates and returns a pid; returns -1 if if unable to allocate a pid (all pids are in use)
- void release_pid(int_pid) – Releases a pid.

NOTE: This programming problem will be modified later in Chapters 4 and 5.