assignment	Problem Set 3
by	Rebecca Turner
course	COSI 21a
instructor	Dr. Liuba Shrira
due	2018-10-20

Note: This example document is provided to demonstrate the capability and visual style of the `brandeis-problemset` document class. The solutions below are not guaranteed to be correct, complete, or relevant.

The source code for this document is available at

```
/macros/latex/contrib/brandeis-problemset/example.tex
```

on CTAN (or, if you have `brandeis-problemset` installed, in your TeX distribution's documentation directory).

Contents

Part I

Textbook problems

Problem 1

An assembly language program implements the following loop:

```
1  int A[51];
2  int i = 1;
3  while(i <= 50) {
4      A[i] = i;
5      i++;
6  }
```

The array of integers A is stored at memory location $x + 200$, where x is the address of the memory location where the assembly program is loaded. Write the assembly program using the assembly language introduced in class.

For a completely unrelated problem, see problem 5 (this is an example of a \label / \ref pair).

```
x + 0               LOAD   R1, $200         ; A = (program location) + 200
x + 4               LOAD   R2, =1           ; i = 1
x + 8   LOOP:       STORE  R2, @R1          ; *A = i
x + 12              ADD    R1, =4           ; A++
x + 16              INC    R2               ; i++
x + 20              BLEQ   R2, =50, LOOP    ; Ensure i <= 50
x + 24              HALT
```

Problem 1.11

Direct memory access is used for high-speed IO devices in order to avoid increasing the CPU's execution load.

1. How does the CPU interface with the device to coordinate the transfer?

2. How does the CPU know when the memory operations are complete?

3. The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of user programs? If so, describe what forms of interference are caused.

1. The CPU sets up "buffers, pointers, and counters for the IO device" and then ignores the transaction entirely; because DMA transfers don't involve the CPU at all, they're especially efficient because they don't saturate the CPU bus.

2. The device controller sends a CPU interrupt when each block of data finishes transferring.

3. A DMA transfer only interferes with user programs as much as any other IO operation might, i.e. the program may not be able to complete other meaningful work before the transfer finishes. From the user's perspective, a DMA transfer is indistinguishable from any other type of IO operation.

   Additionally, a DMA takes a lock on RAM; while a DMA transfer is in progress, no other processes may access RAM, which can be extremely limiting.

Problem 2

In the following, use either a direct proof for the statements (by giving values for c and n_0 in the definition of big-O notation) or cite the rules given in the lecture notes.

1. $\max(f(n), g(n))$ is $O(f(n) + g(n))$. Assume that $f(n)$ and $g(n)$ are non-negative for $n > 0$

2. If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then the product $d(n) \cdot e(n)$ is $O(f(n) \cdot g(n))$

3. $(n + 1)^5$ is $O(n^5)$

4. n^2 is $\Omega(n \log n)$

5. $2n^4 - 3n^2 + 32n\sqrt{n} - 5n + 60$ is $\Theta(n^4)$

6. $5n\sqrt{n} \cdot \log n$ is $O(n^2)$

An example equation which defines e:

$$\exists! e \in \mathbb{R} \text{ such that } \int_1^e \frac{1}{t} dt = 1. \tag{1}$$

The definition of the Mandelbrot set:

$$c \in \mathbb{C}, \ z_0 = 0,$$
$$\lim_{n \to \infty} z_n = z_{n-1}^2 + c \neq \infty \implies c \in \mathcal{M} \tag{2}$$

Problem 3

What do the following two algorithms do? Analyze its worst-case running time and express it using big-O notation.

```
1  Foo(a, n)
2      Input:  two integers, a and n
3      Output: a^n
4      k  ←  0
5      b  ←  1
6      while k < n do
7           k  ←  k + 1
8           b  ←  b * a
9      return b
```

```
1  Bar(a, n)
2      Input:  two integers, a and n
3      Output: a^n
4      k  ←  n
5      b  ←  1
6      c  ←  a
7      while k > 0 do
8          if k mod 2 = 0 then
9               k  ←  k/2
10              c  ←  c * c
11         else
12              k  ←  k - 1
13              b  ←  b * c
14     return b
```

Foo(a, n) computes a^n, and will run in $O(n)$ time always.

Bar(a, n) *also* computes a^n, and runs in $O(\log n)$ time — this is referred to as exponentiation by squaring.

Part II

Scheduling

Problem 5.4

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

| Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

1. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum $= 1$).

2. What is the turnaround time of each process for each of these scheduling algorithms?

3. What is the waiting time of each process for each of the scheduling algorithms?

4. Which of the algorithms results in the minimum average waiting time (over all processes)?

1. SJF

Average wait $= 3.2$.

Process	Turnaround	Waiting
P_1	19	9
P_2	1	0
P_3	4	2
P_4	2	1
P_5	9	4

1	1	2	5	10
P_2	P_4	P_3	P_5	P_1
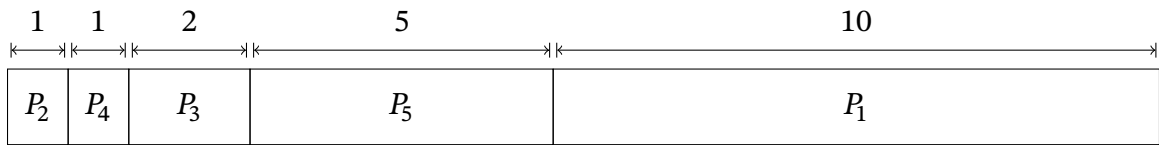
Problem 4

Write a Scheme procedure to calculate an arbitrary up-arrow $a \uparrow^n b$.

```scheme
;;; (up-arrow 2 3 4) = 2^^^4
(define (up-arrow a n b)
  (cond ((= n 1) (expt a b))
        ((and (>= n 1) (= b 0)) 1)
        (else (up-arrow a
                        (- n 1)
                        (up-arrow a n (- b 1))))))
```