Pset 6 Notes (CSCI E-10B)

General Information

a. Problem set 6 is due Monday, December 18, 2023 at 9:00 AM Eastern Time. **The late submission window for problem set 6 (and for the term project) closes on Wednesday, December 20 at 9:00 AM because of semester-end Harvard administration schedules for reporting final grades!** You can resubmit your assignment as often as desired, but each submission's zip must include every file that you want to be graded.

b. We will deduct 10% for a homework assignment that is turned in up to 3 days late. 20% will be deducted if the homework is more than 3 days late. No homework will be accepted more than 7 days late. The last submission controls the late penalty for the entire assignment.

c. Don't procrastinate! Note, though, that the pset covers material from lectures 13 and 14 so you will probably need to defer some questions until you've viewed future lectures and attended or viewed future sections.

d. QtSpim must be able to load your MIPS programs. Programs that QtSpim cannot load will automatically have their score divided by two.

e. Your programs must behave as specified. Do everything the specs say to do. Do not do more than the specs say to do. Precisely follow the directions in the pset (e.g. use the exact filenames specified in the pset), except make adjustments specified by the staff in these notes or on the Ed discussion board.

f. Some of the work is designated as "extra credit." In this course, extra credit points are kept separate from regular credit points. They only come into play at the end of the semester when Dr. Leitner is assigning final grades. If you are on the cusp between, say, a B+ and an A-, extra credit points can influence that decision.[1]

   If the pset says "identify which problem(s) you want treated as extra credit," ignore that! We automatically allocate points to maximize your regular credit score because that contributes the most to your grade in the course.

   If your submission gets a late penalty, then you won't get any extra credit points for that submission.

g. This course emphasizes programming style. Inadequately-commented or -styled programs will lose points. For MIPS programs, please achieve consistent column alignment so that your programs are easier to decipher.

Pencil-and-Paper Exercises

This assignment has no Pencil-and- Paper Exercises.

Programming Problems

Submit complete and correct programs that behave exactly as specified, except make adjustments specified by the staff in these hints, or via email, or on Ed.

Your programs' interaction with the user should match what's in the assignment, and when sample cases are shown in the problem set, your program must produce the same output when given the sample case's input.

[1] The decision is also influenced by your teaching assistant's feedback, so it's a good idea to get to know that person.

a. Problem 1 (**smallestlargest.asm**): [2] Create a MIPS program that prints the smallest and largest values found in a non-empty table of **N** word-sized integers. The location of the first entry in your table should be labeled **table**, and the **.word** statement defining **N**'s value should be labeled **N**. Your submission needs to demonstrate your program with two data sets. To run against one of the data sets, comment out the other data set.

```
N:      .word  9
table:  .word  3, -1, 6, 5, 7, -3, -15, 18, 2
#N:     .word  1
#table: .word  3
```

In section I'll demonstrate a table-processing technique that identifies the end of a table via a label assigned to the memory location following the table's last entry. You cannot use that technique in this problem! Instead, your table-processing loop must be controlled by the value stored at the memory location with the label **N**.

b. Problem 2 (**palindrome.asm**): Modify the **palindrome.asm** [3] program so that it ignores whitespace, capitalization and punctuation. Do not "preprocess" the string outside of the palindrome-detection loop. Instead, modify the code in **palindrome.asm** so that the palindrome-detection loop correctly ignores whitespace, ignores punctuation, and deals with upper- and lower-case characters.

c. Problem 3 (**perfect.asm**): [4] This problem is required for grad students. It is extra credit for other students. There are three perfect [5] numbers between 5 and 500. Write a MIPS program that will test all integers between 5 and 500 inclusive and print out the "perfect" ones.

For 3 points of extra credit, construct your solution in such a way that the main program repeatedly calls on a subroutine named **perfect**. This subroutine is passed a single integer argument that it checks for the property of being "perfect." A true/false answer of some sort gets returned by this subroutine.[6]

---

[2] The problem set did not specify a filename for problem 1. Please use the name **smallestlargest.asm**.

[3] Located on the Java Resources page.

[4] The problem set did not specify a filename for problem 3. Please use the name **perfect.asm**.

[5] A "perfect number" is a positive integer greater than 1 which is equal to the sum of its divisors (except for the original number itself). For example, 6 is perfect because
(a) 1, 2 and 3 are the only integer values smaller than 6 that evenly divide into 6 and
(b) 6 is equal to 1 + 2 + 3
On the other hand, 12 is not perfect, since 12 ≠ 1 + 2 + 3 + 4 + 6.

[6] For instance, your subroutine might return the value **0** to indicate **false** (the argument is not a perfect number) and it might return the value **1** to indicate **true** (the argument is a perfect number).