CS61C Spring 2010 Midterm Rubric

Question 1 - Brian Harvey would be so proud... (C) - 20pts

- a) 6pts
    - 1pt for error checking (2 lines of code)
        - if they don't have a malloc, only need one error check
        - only need to check malloc calls, not inputs since problem says "with the given main code"
    - 1pt per line for rest (5 lines in solution)
    - no point for malloc'ing a word if it gets put into the new node (to stay consistent >_>)
- b) 4pts - 1pt each
    - if static and code have 4 4 or 8 0, give 2 points
    - elseif static is 4 or 8, or code is 4 or 0, give 1 point
- c) 2pts
    - missing a go!, -1
- d) 2pts
    - [main->]se->share_or_new->share_or_new[->error]
        - square brackets optional for full points
        - if missing one share_or_new give 1 point still just for realizing there's an error
    - 0 if there's a box&ptr diagram
- e) 6pts - 2pts each (excluding the malloc check)
    - for small errors (missing ! or == 0 in strcmp, off by 1 or constant malloc size) -1
    - -1 if they mentioned malloc but didn't specify size

Question 2 - L1 and L2 below are Booth needed for the algorithm... (MIPS->C) - 10pts

- a) 4pts
    - Case A (4pts): multiplies $a0 and $a1 and puts the product mod 2^32 in $v0
    - Case B (3pts): multiplies $a0 and $a1 and puts the product in $v0
    - Other ways to get points
        - If they have a correct sentence/sum notation mod 2^32 (3pts)
        - If they have a correct sentence/sum notation (2pts)
- b) 2pts
    - Case A
        - 2/2 - Yes, two's complement numbers still add correctly
        - 1/2 - No, because sll can break sign of result
    - Case B
        - 2/2 - No, because sll can break sign of result
    - Common to both
        - 2/2 - Yes, two's complement numbers still add correctly
        - 2/2 - Yes, the answer is congruent mod 2^32
        - 1/2 - Some true statement that is consistent with your response to part a
- c) 2pts
    - 2/2 - MSB of $a0 is 0 (or some equivalent statement)
    - 1/2 - MSB of $a0 is 0 and MSB of $a1 is 0 (or some equivalent statement)
- d) 2pts
    - 2/2 - infinite loop, nothing for anything else

Question 3 - Tasting Menu - (Potpourri) - 18pts

- a) 2pts - 1 pt for an off-by-one error where the 0 was miscounted (2^(n+1), 2(2^n-1)-1, etc)
- b) 2pts - Mostly all or nothing. 1 pt for getting the inversion right but leaving "constant offset" blank.
- c) 1pt - All or nothing.
- d) 3 pts
    - 2 pts if FIRST or NEXT is in wrong box, the blocks of memory weren't labeled (F,N,B still correctly placed), or some other very minor issue.
    - 1 pt if only 2 of the 3 allocations are different. 1 pt if the position box was left entirely blank but C code was correct.
- e) 8 points.
    - 1 pt per explanation, only accepting answers from the answer key. If they list an additional explanation that isn't completely wrong, ignore it.
    - 3 pts per MIPS box, 1 pt per line. -1 pt total if mostly correct but non-minimal (includes saving $ra) or if very similar mistake is made in both boxes. Ignore register spilling (don't subtract point for non-minimal). We allowed pseudoinstructions for the multiplication, although a shift was preferable.
- f) 2pts - 1 pt for listing all R-type instructions (2^11), or where the problem breakdown was correct (new R-type + I-type + J-type), but one of the categories was just slightly off.

Question 4 - Did somebody say "Free Lunch"?! (Float) - 11pts
- All or nothing for all questions.
- If both blanks have answers but none are circled, assume they circle the smaller number.
- If one blank have answer but it is not circled, assume they circle that number.
- If same answer for both blank, and one of them is right, give 1 point.
-
- a) 2pts
- b) 2pts
- c) 2pts
- d) 2pts
- e) 2pts
- f) 1pt

Question 5 - Euclid's Revenge... (MIPS) - 15pts
Common Mistakes:
- Using mod as an instruction: 6/10 on the MIPS part
- Forgetting to save $a1: 6/10 on the MIPS part
- Circling addiu $sp, $sp for the instruction that crashes

- a) 10pts
    - 2pts for base case
        - 1 pt for each line
    - 2pts for prologue
        - 1 pt for saving $ra
        - 1 pt for saving $a1
    - 4pts for body (mod & recursive call)
        - 2 pt for calling mod correctly [some students might think there's a mod MIPS inst]
            - I.e., mod $s0, $a0, $a1
            - I didn't see many students do this - maybe 2 in my 50
            - You still need to know how to deal with it

      - I had one question out of the three downstairs DSP students.
      - These students won't have to worry about jal, saving ra, anything
      - It fundamentally changes the difficulty of the question
      - I don't have an easy answer to this, it's an honest mistake ... we say "MIPS subroutine", but it's not nec clear that a subroutine and inst are different!
      - Which means we have no feedback whether they know how to handle the problems that jal mod creates. Giving points means they DO know, taking points means they DON'T, but we have no information!
      - We still need the pro/epilogue to save $ra,
      - Why don't you start grading based on the rubric and save the ones that are hard that we should talk about... what are your timing constraints?
      - spc at 1
      - Your spec looks good so far; I'd say we take a look at the problem cases we see above and give the half the points (vis-a-vis mod), it's easier. So the 10-pt question (from 10 miles up) should probably get 5 pts. How does that feel?
      - maybe half of the prologue-body-epi, I think they should still get full for the base case if they did that right. Right, full base case, I'm ok with 5 or 6 pts total for the MIPS part for those who didn't do mod right. ok? ok
      - If you guys need to conduct a dialogue here you can insert comments... -Michael Greenbaum 3/10/10 2:55 AM
    - 2 pts for setting up $a0 and $a1 correctly for recursive GCD call
  - 2pts for epilogue
    - 1 pt for restoring $ra
    - 1 pt for restoring $sp
    - Only earn these points if prologue is correct?
  - - x pts for "minor" errors (bad mips instructions, ...
    - saw a few non existant MIPS inst, like set
      - that should be ok, since I said they can make up their own MAL
      - but it has to work
  - -1 pt for not being efficient
- b) 1pt
  - .5 if showed work but wrong final answer ??
    - i.e., converting to hex wrong, but set up the field correctly. takes longer to grade though....
    - I don't like 1/2 points. It's only 1 point. (if it were 10 pts, I could see it)
    - HOWEVER if they write the first instruction incorrectly, their answer could still be correct (e.g., beq instead of bne) and they should get full credit here!
    - ok
- c) 1pt
  - all or nothing
- d) 2pts
  - 1 pt for circling the right line (sw)
    - The system NEVER checks for registers, $sp could be set to a random # but that's not an error

        - The system only flags when memory is being accessed when it can't (i.e., the sw request has crossed into the heap), which might be unclaimed. imho, it's possible that the stack could overwrite the heap and it not cause an error, if the user has that area as part of a previous malloc. But that's the error, sw
        -
    - 1 pt for stack overflow (or something VERY close, use your judgement obv)
    - Long correctly points out that GCD dramatically reduces the size of its inputs, however if the heap is very full it can still have a stack overflow
        - so...all or nothing...?
        - Yes, all or nothing. GCD could have only *3* recursive calls, but it could still crash because of stack overfl and the answer is STILL to rewrite iteratively.
        - Use your judgement on how close they write to what we want
- e) 1pt
    - 1 pt for saying iterative solution