Grading Guidelines for Part-1 of Lab Test 3

Problem Set 1

Grade this part strictly, do not be too lenient. For 1 mark subparts, if they are not close, give 0.

If they do not submit the Intermediate file and were not excused by us: deduct 0.5 for

- Output: 2 marks
  - No marks for printing the polynomials
  - Exact message format not important.
  - For each term, enter degree first, then coefficient. They have been clearly told to do so in the question. If it does not run, it does not run.
  - Enter inputs exactly in this order, we want to test that they considered degrees entered in any order
  - Test case 1: n = 4, P = (3, 6), (1, 2), (5, 3), (7,1) (so P = $x^7 + 3x^5 + 6x^3 + 2x$)
    m = 3 Q = (0, 4), (4, 1), (5, 2) (so Q = $2x^5 + x^4 + 4$)
    - Sum of polynomials: $x^7 + 5x^5 + x^4 + 6x^3 + 2x + 4$ (0.5 marks, binary)
    - Product of polynomials: $2x^{12} + x^{11} + 6x^{10} + 3x^9 + 12x^8 + 10x^7 + 4x^6 + 14x^5 + 24x^3 + 8x$ (0.5 marks, binary)
  - Test case 2: n = 4, P = (4, 1), (1, 2), (5, 3), (7,1) (so P = $x^7 + 3x^5 + x^4 + 2x$)
    m = 3, Q = (1, -2), (4, -1), (5, -3) (so Q = $-3x^5 - x^4 – 2x$)
    - Sum of polynomials: x^7 (0.5 marks, binary)
    - Product of polynomials: $-3x^{12} - x^{11} - 9x^{10} - 6x^9 - 3x^8 - 12x^6 - 4x^5 - 4x^2$ (0.5 marks, binary)
  - For the first print of polynomials, they can print the terms in any order, but the prints after the add and multiply must be in decreasing order of degree.
- Code: 8 marks
  - AddPoly() function: 5 marks
    - For this, since they do not know a-priori how many terms will be there in the sum (terms present may even cancel out due to negative coefficients), and they cannot use any additional array, they must first do a dummy addition to count the number of terms with non-zero coefficients, allocate that sized array inside the polynomial to be returned, store the sums there, and actually sort there in the array within the polynomial. Note that since coefficients can be negative, the count cannot simply be the maximum degree in the polynomials, as some terms may disappear during the summation. Also, they cannot sort P1 and P2 as they should remain unchanged as per the question.
    - The POLY structure and the array inside it must be malloc'ed. Static allocations will not work as it is free'd after the function returns. Global is not allowed.
    - Marks Distribution

        - Malloc’ing the POLY structure for return: 1 mark
            - Give 0 in this part if they use a static array and return a pointer to it.
        - Addition of polynomials: 3 marks
            - Counting exact number of terms in sum: 1 mark
                - Deduct 0.5 if they do any logic that does not take care of possible negative coefficients and terms disappearing.
                - Give 0 if they have not considered BOTH the possibility that terms in P1 may be missing in P2 AND vice-versa. Several codes have only checked one way as I sampled.
            - Allocating array inside POLY to be returned and computing and storing terms of sum: 2 marks
            - Deduct 1 if they assume the degrees are in sorted order in t_list array of POLY P1 and P2
            - Deduct 1 if they sort P1 and P2 first
        - Sorting of degrees at end: 1 mark
            - Binary. 0 or 1 depending on whether the sort is correct or not, including not using any static array. Can use any sorting method.
    - Deduct 2 if they use any additional array anywhere (other than the dynamically allocated array inside the POLY structure to be returned).
- MultPoly() function: 3 marks
    - Unlike in AddPoly(), they are allowed to use additional dynamically allocated arrays here. So they can find the size (or maximum bound for it) to malloc an array to store theproduct, then sort it, and then write it back in the POLY structure to return. The number of elements in this array can be anything based on the degrees of P1 and P2, but must not be a constant. Also, since the POLY structure’s array must be of exactly the size of the number of elements, they have to count the terms with non-zero coeffs after computing the product and malloc that array only after that.
    - Malloc’ing the POLY structure for return: 0.5 marks
            - Give 0 in this part if they use a static array and return a pointer to it.
    - Computing Product: 2 marks
        - Deduct 1 mark if they use any static array
    - Sorting of degrees at end: 0.5 marks
        - Binary (0 or 0.5) depending on sort is correct or not
    - Deduct 1 if they assume the degrees are in sorted order in t_list array of POLY P1 and P2
    - Deduct 1 if they sort P1 and P2 first

- No marks for main(), **BUT deduct 2 if they sort P and Q before calling the functions** (as that makes both Add and Mult easier). But then do not deduct for sorted P1 and P2 inside Add and Multiply again
- Deduct 1 mark if they write any other functions for print, sort etc. Clearly told not to write any. Just wanted them to type a lot of code, or at least know they can cut-and-paste it.

Problem Set 2

Grade Leniently. Use your judgement.

- Output: 1 mark
  - Test case 1: P = $x^7 + 3x^5 + 6x^3 + 2x + 1$, Q = $2x^5 + 7x^4 + 4$
    - Coefficient of 3rd Largest degree in product = 6 (0.5 marks, binary)
    - Minimum missing degree = 2 (0.5 marks, binary)
- Code: 5 marks
  - Find3rdMax() function: 2 marks
    - Simple product computation, just need to keep track of 3rd highest degree terms (all positive coefficients so easy)
    - Basic loops for product computation: 1 mark
    - Keeping track of 3rd Max: 1 mark
      - Deduct 0.5 if they didn't leave out zero-coefficient terms
    - Deduct 1 if they use any additional array (cannot first compute the product then do)
    - Deduct 0.5 if they do not consider the case where here is no highest degree term
  - FindMissing function: 2 marks
    - Loops to find missing terms: 1 mark
      - Since they cannot use any additional array, they may try out each degree and find its coefficients in the product term, so 3 loops
    - Keeping track of minimum: 1 mark
    - Deduct 0.5 if they do not consider the case where there is no missing term
    - Give 0 in this part if they use any additional array (cannot first compute the product then do)
  - main(): 1 mark
    - reading and printing the polynomials – 1 mark