Indian Institute of Technology, Kharagpur

CS19001 Programming and Data Structures Laboratory, Autumn 2019

Assignment for Week 9 (October 28, 2019)

Total Marks: 40 **Submission Deadline:** 17:45

INSTRUCTIONS

1. Submit a single C file named **[rollno].c** where '[rollno]' is your roll number.
2. You may consult your notes, books or manual pages.

PROBLEMS

The purpose of this assignemnt is to build a library to perform certain computations on *sparse polynomials*. A univariate polynomial of degree d in an indeterminate x is usually represented as $f(x) = \sum_{i=0}^{d} c_i x^i$ which is also called the *dense* representation. If many of the c_i's are 0, it is natural to consider a *sparse* representation that only records the non-zero terms. In other words, if n terms are non-zero then we have

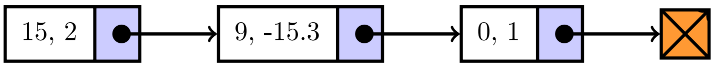
$$f(x) = \sum_{j=1}^{n} c_{\alpha_j} x^{\alpha_j}$$

where $0 \leq \alpha_i \leq d$ are precisely the exponents of the n terms with non-zero coefficients c_i. An efficient representation of a sparse polynomial is a linked list of nodes containing two variables – coefficient and exponent – with the nodes sorted in decreasing order of exponent. Below is a suggested definition of a node.

```
typedef struct _node{
   int exp;
   float coeff;
   struct _node *next;
} node;

typedef node *poly1, *poly2;
```

For example, the polynomial $2x^{15} - 15.3x^9 + 1$, would be stored as



Define a polynomial (node) as described above .

1. Write a function `node* insert(node *poly, int exp, float coeff)` to insert a new node into a list so that the descending order of exponents is maintained and returns the modified list.
2. Write a function `void print_poly(node *poly)` that prints the polynomial represented by `poly`.
3. Write a function `float eval_poly(node *poly, float x)` that evaluates the polynomial represented by `poly` at a point `x` and returns the resulting value.

4. Describe two functions `node* add_poly(node *poly1, node* poly2)` and `node* multiply_poly(node *poly1, node* poly2)` that take two polynomials as input and outputs the sum and product (respectively) of the two polynomials. Ensure that the returned polynomial has a similar representation (with exponents in descending order) and contains at most one node with a particular exponent.

Submit a single C file. You may display the following tasks to the user and ask for a choice:

- Evaluate a polynomial
- Add two polynomials
- Multiply two polynomials

Lists are populated by user defined values. For instance, when the user chooses to add two polynomials, then two polynomials must be read from the user and corresponding lists must be created according to the values read. Then using the function `add_poly` their sum should be computed and the resulting polynomial printed using `print_poly`.

Practice Question: Try division (computing quotient and remainder). This is not meant for submission.

Sample Input/Output 1

```
1. Evaluate a polynomial
2. Add two polynomials
3. Multiply two polynomials
What would you like to do? 3

Enter (exponent, coefficient) pairs for the first polynomial.
To stop enter (-1,0):
2 3
9 1
0 1
-1 0

Enter (exponent, coefficient) pairs for the second polynomial.
To stop enter (-1,0):
0 4
2 5
-1 0

The two polynomials you entered are
x^9 + 3 x^2 + 1
5 x^2 - 4
Their product is
5 x^11 - 4 x^9 + 15 x^4 - 7 x^2 - 4
```

Sample Input/Output 2

```
1. Evaluate a polynomial
2. Add two polynomials
3. Multiply two polynomials
What would you like to do? 1

Enter (exponent, coefficient) pairs defining the polynomial.
To stop enter (-1,0):
3 2
2 -6
1 2
0 -1
-1 0

Enter the value at which the polynomial must be evaluated: 3
The polynomial 2 x^3 - 6 x^2 + 2 x - 1 evaluates to 5 at 3.
```