EE2410 Data Structure Coding HW #1 (Chapter 1~2 of textbook)

due date 4/2/2024 23:59

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program, i.e., write a client main() to demonstrate all functions you designed using example data.

Submit your homework before the deadline (midnight of 4/2). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have SERIOUS consequences.

Arrays: **due date: 23:59, 4/2/2024 (Tue.)**

1. (30%)

Write a C++ program to implement the **ADT2.3 Polynomial** below using Representation 3 (dynamic array of (coef, exp) tuples).

```
class Polynomial;
class Term{
friend Polynomial;
private:
   float coef;
   int    exp;
};
class Polynomial {
// p(x) = a0 x^e0 + … + an x^en
// where ai is nonzero float and ei is non-negative int
public:
      Polynomial( );
      //construct the polynomial p(x) = 0
      ~Polynomial();
      //destructor
      Polynomial Add(Polynomial poly);
      //return the sum of *this and poly
      Polynomial Subt(Polynomial poly);
      //return the difference of *this and poly
      Polynomial Mult(Polynomial poly);
      //return the product of *this and poly
      void NewTerm(const float theCoeff, const int theExp);
      float Eval(float f );
```

```
    //Evaluate the polynomial *this at f and return the results
    int operator!();
            // if *this is the zero polynomial, return 1; else return 0;
    float Coef(int e);
             // return the coefficient of e in *this
    int LeadExp();
            // return the largest exponent in *this
    friend ostream& operator<<(ostream& os, Polynomial& p);
    friend istream& operator>>(istream& is, Polynomial& p);
  private:
    Term* termArray;
    int   capacity;
    int   terms;
};
```

Show the results of one runs of your program (execution trace) to **demonstrate** all the operations (Add, Subt, Mult, Eval, !, Coef, and LeadExp) as well as input, output functions as follows:

```
int main(){
    construct polynomial objects a, b
    use >> to build polynomial object a = 2x^3 + 3x^2 + 4x + 5, b = x^3 – x^2 + x – 1
    demo   <<
    demo   << results of Add, Subt, Mul
    demo results of a.Exal(1), b.Eval(2), a.Coef(5), b.LeadExp
}
```

sol:

Execution trace 1:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.1_polynomial"
Enter the first polynomial f1(x) in the form of coef exp coef exp ...
1 3 2 2 1 1 4 0
Enter the second polynomial f2(x) in the form of coef exp coef exp ...
1 2 2 1 11 0
===================================
f1(x) = (1)*x^(3) + (2)*x^(2) + (1)*x^(1) + (4)
exp of the leading term: 3
coef of the leading term: 1
f1(x) is not zero
===================================
f2(x) = (1)*x^(2) + (2)*x^(1) + (11)
exp of the leading term: 2
coef of the leading term: 1
f2(x) is not zero
===================================
f1(x) + f2(x) = (1)*x^(3) + (3)*x^(2) + (3)*x^(1) + (15)
f1(x) * f2(x) = (1)*x^(5) + (4)*x^(4) + (16)*x^(3) + (28)*x^(2) + (19)*x^(1) + (44)
f1(x) - f2(x) = (1)*x^(3) + (1)*x^(2) + (-1)*x^(1) + (-7)
===================================
Enter a value v: 2
f1(v) + f2(v) = 41
f1(v) * f2(v) = 418
f1(v) - f2(v) = 3
===================================
Enter a exp to see its coef of f1(x) * f2(x): 2
The corresponding coef is: 28
sunpierce@pierces-MacBook-Air output %
```

Execution trace 2:

This is a particular case showing how program deals with zero polynomial.

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.1_polynomial"
Enter the first polynomial f1(x) in the form of coef exp coef exp ...
1 2 1 1 1 0
Enter the second polynomial f2(x) in the form of coef exp coef exp ...
0 0
===================================
f1(x) = (1)*x^(2) + (1)*x^(1) + (1)
exp of the leading term: 2
coef of the leading term: 1
f1(x) is not zero
===================================
f2(x) = 0
exp of the leading term: 0
coef of the leading term: 0
f2(x) is zero
===================================
f1(x) + f2(x) = (1)*x^(2) + (1)*x^(1) + (1)
f1(x) * f2(x) = 0
f1(x) - f2(x) = (1)*x^(2) + (1)*x^(1) + (1)
===================================
Enter a value v: 2
f1(v) + f2(v) = 7
f1(v) * f2(v) = 0
f1(v) - f2(v) = 7
===================================
Enter a exp to see its coef of f1(x) * f2(x): 8
The corresponding coef is: 0
sunpierce@pierces-MacBook-Air output %
```

2. (35%)

Write a C++ program to implement the **ADT2.4 SparseMatrix** in textbook shown

below.

```
class SparseMatrix
{//三元組，<列，行，值>，的集合，其中列與行為非負整數，
 //並且它的組合是唯一的；值也是個整數。
public:
    SparseMatrix(int r, int c, int t);
    //constructor.
    //r is #row, c is #col, t is #non-zero terms
    SparseMatrix Transpose( );
    SparseMatrix FastTranspose( );
    //回傳將 *this 中每個三元組的行與列交換後的 SparseMatrix
    SparseMatrix Add(SparseMatrix b);
    // 如果 *this 和 b 的維度一樣，那麼就把相對應的項給相加，
    // 亦即，具有相同列和行的值會被回傳；否則的話丟出例外。
    SparseMatrix Multiply(SparseMatrix b);
// 如果*this 中的行數和 b 中的列數一樣多的話，那麼回傳的矩陣 d= *this 和 b
//（依據 d[i][j]=Σ(a[i][k]・b[k][j]，其中 d[i][j]是第 (i,j) 個元素）相乘的結果。
// k 的範圍從 0 到*this 的行數減 1；如果不一樣多的話，那麼就丟出例外。
    //other needed functions
};
```

You should build you program based on the example codes in the book and implement the **Add** function and functions to **input**, **output** a sparse matrix by **overloading** the **>>** and **<< operators**.

You should try out at least one runs of your program to demonstrate the Add, Mult, FastTranspose, and input, output functions.

```
int main(){
    use >> to build sm object a(4x3, 4 terms), b(4x3, 5 terms), c(3x3, 4 terms)
    demo   <<
    demo   << results of Transpose, Fast Transpose, Add, Mul
}
```

sol:

Execution trace 1:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.2_sparsematrix"
For matrix M1:
Enter the rows in total: 4
Enter the cols in total: 4
Enter the matrix's row col val row col val ...
0 0 1 1 2 8 2 1 6 3 2 4
M1 = { (0,0,1) (1,2,8) (2,1,6) (3,2,4) }
For matrix M2:
Enter the rows in total: 4
Enter the cols in total: 4
Enter the matrix's row col val row col val ...
0 0 7 1 3 4 2 1 5 3 1 6
M2 = { (0,0,7) (1,3,4) (2,1,5) (3,1,6) }
==================================
Fast Transpose of M1 = { (0,0,1) (1,2,6) (2,1,8) (2,3,4) }
Transpose of M1 = { (0,0,1) (1,2,6) (2,1,8) (2,3,4) }
Fast Transpose of M2 = { (0,0,7) (1,2,5) (1,3,6) (3,1,4) }
Transpose of M2 = { (0,0,7) (1,2,5) (1,3,6) (3,1,4) }
==================================
M1 * M2 = { (0,0,7) (1,1,40) (2,3,24) (3,1,20) }
M1 + M2 = { (0,0,8) (1,2,8) (1,3,4) (2,1,11) (3,1,6) (3,2,4) }
sunpierce@pierces-MacBook-Air output %
```

Execution trace 2:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.2_sparsematrix"
For matrix M1:
Enter the rows in total: 2
Enter the cols in total: 2
Enter the matrix's row col val row col val ...
0 0 2 0 1 3 1 0 5
M1 = { (0,0,2) (0,1,3) (1,0,5) }
For matrix M2:
Enter the rows in total: 2
Enter the cols in total: 2
Enter the matrix's row col val row col val ...
0 1 4 1 0 5 1 1 6
M2 = { (0,1,4) (1,0,5) (1,1,6) }
==================================
Fast Transpose of M1 = { (0,0,2) (0,1,5) (1,0,3) }
Transpose of M1 = { (0,0,2) (0,1,5) (1,0,3) }
Fast Transpose of M2 = { (0,1,5) (1,0,4) (1,1,6) }
Transpose of M2 = { (0,1,5) (1,0,4) (1,1,6) }
==================================
M1 * M2 = { (0,0,15) (0,1,26) (1,1,20) }
M1 + M2 = { (0,0,2) (0,1,7) (1,0,10) (1,1,6) }
sunpierce@pierces-MacBook-Air output %
```

3. (35%)

Write a C++ program to implement the **ADT2.5 String**.

```
class String
{
public:
    String(char *init, int m);
    // constructor using input string init of length m
    bool operator == (String t);//equality test
    bool operator!( ); // empty test, true or false
    int Length( );//get the number of characters of *this
```

```
    String Concat(String t);
    // concatenation with another string t
    String Substr(int i, int j);    // generate a substring i~i+j-1
    int Find(String pat);
    int FastFind(String pat);
    // Return an index i such that pat matches the substring
    // of the object begins at position i.    Return -1 if pat
    // is empty or not a substring of the object
    String Delete(int start, int length); //remove length characters beginning
    // at start
    String CharDelete(char c); //returns the string with all occurrence of c
    //removed.
    int Compare(String y); //compare two strings of letters of alphabet.
    //return -1 if <y, 0 if =y, and 1 if >y.
    //If two strings of letter of alphabet, x = (x0,…,xm-1) and y=(y0,…,yn-1)
    //where xi, yj are letters, then the Compare member function will decide
    //whether x<y, x=y, or x>y, where x < y means if xi=yi for 0≤i<j and xj<yj
    //or if xi=yi for 0≤i≤m and m<n. x=y means m=n and xi=yi for 0≤i<n. x>y
    //means if xi=yi for 0≤i<j and xj>yj or if xi=yi for 0≤i≤n and m>n.
}
```

You should try out at least two example runs of your program to demonstrate all those functions.

sol:

Execution trace 1:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.3_string"
Enter the length of the string: 10
Enter the input string: abccddabcd
Enter the length of the string: 8
Enter the input string: ddabcdac
String 1 is : abccddabcd
String 2 is : ddabcdac
========================================
String 1 is not null
String 2 is not null
========================================
String 1 is smaller than String 2
========================================
String 1 concate with String 2 :abccddabcdddabcdac
========================================
Enter the value i, j to get i~i+j-1 of String 1: 3 4
Result: cdda
========================================
Suppose the pattern is the above result.
Then the position where the pattern starts in String 2:
By the naive method, the result is: -1
By the KMP method, the result is: -1
========================================
Enter the value i, j to delete i~i+j-1 of String 1: 2 3
Result: abdabcd
========================================
Choose a letter to delete from String 2: d
Result: abcac
sunpierce@pierces-MacBook-Air output %
```

Execution trace 2:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_1/output"
sunpierce@pierces-MacBook-Air output % ./"1.3_string"
Enter the length of the string: 15
Enter the input string: computerscience
Enter the length of the string: 11
Enter the input string: datascience
String 1 is : computerscience
String 2 is : datascience
========================================
String 1 is not null
String 2 is not null
========================================
String 1 is smaller than String 2
========================================
String 1 concate with String 2 :computersciencedatascience
========================================
Enter the value i, j to get i~i+j-1 of String 1: 8 7
Result: science
========================================
Suppose the pattern is the above result.
Then the position where the pattern starts in String 2:
By the naive method, the result is: 4
By the KMP method, the result is: 4
========================================
Enter the value i, j to delete i~i+j-1 of String 1: 2 7
Result: cocience
========================================
Choose a letter to delete from String 2: a
Result: dtscience
sunpierce@pierces-MacBook-Air output %
```