POPL2: Assignment 2

Individual Assignment
Deadline: 14th March 2021
Please post your queries on or before 8th March 2021

Notes:

- Use C++ as the language for this assignment.
- Comment your code
- Submission Guidelines: Zip the “src” folder into a file named “<roll_number>.zip” (example: “cs19btech11000.zip”) and submit it on Google Classroom.
- File structure:
  On running `make` inside the `src` folder, the executables inside the bin folder must be generated. Do not include the generated executables in the submissions.

```
src/
|-bin/ (generated)
| |-1.out (generated)
| |-2a.out (generated - string based)
| |-2b.out (generated - linked-list based)
| |-3.out (generated)
|-Makefile
|-1
| |-*.cpp
| |-...
|-2
| |-*.cpp
| |-...
|-3
| |-*.cpp
| |-...
```

Questions:

1) Define, implement, and test a set of integers, class Intset. Provide union, intersection, and symmetric difference operations.
   - Input:
     - The first line has `n` and `m` - the number of operations (n) and the number of sets(m). The IDs of the sets are {0, 1, ..., m-1}
     - The next `n` lines have an operation which is one of the following:
       (1) A <id> <element> - Add the `element` in the Intset `id`
       (2) U <id1> <id2> - Perform union on the sets `id1` and `id2`. `id1` will be the new ID.
       (3) I <id1> <id2> - Perform intersection on the sets `id1` and `id2`. `id1` will be the new ID.

(4) S <id1> <id2> - Perform symmetric difference on the sets `id1` and `id2`. `id1` will be the new ID.

- Output: The final sets as space separated values. First line has set with ID 0, the second line has set with ID 1 and so on.
- Example:
  - Input:

```
10 3
A 0 1
A 1 2
A 2 3
A 0 4
A 2 5
U 0 1
A 0 3
I 0 2
U 0 1
S 2 0
```

  - Output:

```
2 3
2
2 5
```

2) Define a class for analyzing, storing, evaluating, and printing simple arithmetic expressions consisting of integer constants and the operators +, −, ∗, and /.

   The public interface should look like this:

```
class Expr {
  // ...
  public:
  Expr(const char*);
  int eval();
  void print();
};
```

   The string argument for the constructor **`Expr::Expr()`** is the expression. The function **`Expr::eval()`** returns the value of the expression, and **`Expr::print()`** prints a representation of the expression on **`cout`**. A program might look like this:

```
Expr x("123/4+123*4-3");
cout << "x = " << x.eval() << "\n";
x.print();
```

Define class **`Expr`** twice: once using a linked list of nodes as the representation and once using a character string as the representation. Experiment with different ways of printing the expression: fully parenthesized, postfix notation, assembly code, etc.

- Input: Expression as a string
- Output: `<value>` (to two decimal places)
- Example:
  - Input:
    `123/4+123*4-3`
  - Output:
    `519.75`

3) Write a function that, given an **`istream`** and a **`vector<string> S`**, produces a **`map <string, vector<int>>`** holding each string and the numbers of the lines on which the string appears. Run the program on a text-file with no fewer than 1,000 lines looking for no fewer than 10 words.
   - Input: The first line has the file name for **istream**. The second line has `n` - the number of strings for the **vector<string> S**. The next `n` lines each have one line for **S[i]**
   - Output: On the i-th line, output the vector corresponding to **S[i]** as space separated values.
   - Example:
     - Input:

```
bigfile.txt
5
This is the first line
This is the second line
This is the third line
This is the fourth line
This is the fifth line
```

     - Output:

```
2 4 41 43 124
1 6 8 11 73
12 24
96
100
```

     - Explanation:
       The first line occurs on the line numbers 2, 4, 41, 43 and 124. The second line occurs on line numbers 1, 6, 8, 11, 73 and so on. The line numbers start from line number 1.