Programmer's Guide and Report for Project 2-multiSets

Source:

Header files:
COP3503su14_Proj2_WenlanT_main.h
COP3503su14_Proj2_WenlanT_multiSet.h
COP3503su14_Proj2_WenlanT_object.h

Class implementation files:
COP3503su14_Proj2_WenlanT_multiSet.cpp
COP3503su14_Proj2_WenlanT_object.cpp

Application file:
COP3503su14_Proj2_WenlanT_main.cpp

Makefile:
COP3503su14_Proj2_WenlanT_makefile

Author's name: Wenlan Tian

Date last modified: 06-17-2014

Problem Statement
This program is used to test a system for entering, storing, manipulating and printing sets of various sorts with counts. The program will maintain a current multiset and a current verbose state. There are 17 commands with different functions in this program.

Options
Command line options allow for printing of a help string, for suppression of normal prompts, for verbose prompts and for original input file. In silent mode, outputs 0 or 1 for composite or prime instead of an English sentence. In verbose mode, the user prompt includes a short list of all commands (i.e., the numbers and a one-word description along with parameter if any) and other additional information. In silent mode, there is no prompt and there is no output to the console. Error messages are printed to standard error.

Organization of Code
This program defined two classes: object and multiSet.

Object has two private characters - name and count. To get the information, getters and setters are used for object.

MultiSet is a vector of object. So the name and count in the multiSet can be accessed by multiSet[i].getName() and multiSet[i].getCount(). The getline() function and string stream input were used to get two inputs off of a line from cin or file.

The main loop prompts for user input (if not silent), and reads the user input into a temporary integer variable. There are 0-17 commands. If the input is a number out of this range or not a number, an error message will be printed to console.

Functions, Methods, Procedures

MultiSet has 13 functions:

inputFile()-open and read a list from a file to the current multiSet
reset()-reset current multiSet to the empty set
printToFile()-open and write the current multiSet to a file
print()-print current multiSet to the console
unionSets()-open and union a multiSet from a file with the current multiSet
subtractSets()-open and subtract multiSet from a file from the current multiSet
differenceSets()-open and find the difference between a multiSet from a file and the current multiSet
intersectSets()-open and find the intersection between a multiSet from a file and the current multiSet
findItem()-test if <item name> is in the current multiSet
insertItem()-add <item name> to the current multiset with number <count> if <item name> is not in the current multiset, or increase <item name>'s number by <count> if it is
deleteItem()-remove <item name> from the current set if it is in it
reduceItem()-reduce the number of <item name> by <count>
maxSets()-open and find the maximum open and find the maximum

The main() has the has the “HELP” for a detailed list of commands, and verobseOutput() function including a short list of commands when in verbose state. The main() also has the methods to call different states (verbose, silent and normal) using different combination of bool verbose and bool silent.

Known Bugs
The program cannot distinguish whether item name is a real name or a number. If the input is a number, the program will still run.

User input: when there is more stuff on the command input, the program will use the first one or two to finish the current program. The remaining command input will be considered another command for another program.

Testing
Testing was done by using three files: a, b and c. Script of testing session follows.

```
[a]                  [b]                  [c]
foo 5                foo 4                foo 3
bar 4                bar 4                    bar    45
baz 2                baz 4                foo bar
boz 1                buz 4                bike car 33

                                          banana 18 19
                                          apple
                                          grape    14
```

```
sun01:6% ./proj2
> 1 c
[foo bar] is not in correct format on user input line
```

```
Please make sure the correct format: < item name> <count> per line
[bike car 33] is not in correct format on user input line
Please make sure the correct format: < item name> <count> per line
This line is empty
line malformed: banana 18 19
<apple>'s <count> is missing
New Set Loaded
> 1 a
New Set Loaded
> 8
Current multiset:
foo 5
bar 4
baz 2
boz 1
> 2 a
a union completed
> 8
Current multiset:
foo 10
bar 8
baz 4
boz 2
> 3 b
b subtraction completed
> 8
Current multiset:
foo 6
bar 4
boz 2
> 5 b
b intersetction completed
> 8
Current multiset:
foo 4
bar 4
> 0


sun01:7% ./proj2 -s
1 a
2 b
3 d
Unable to open file: d
13
0 - exit; 1 - input <file>; 2 - union <file>; 3 - subtract <file>;
4 - difference <file>; 5 - intersect <file>; 6 - reset current set
7 - write <file>; 8 - print; 9 - find <item>; 10 - insert
<item><count>;
11 - delete <item>; 12 - reduce <item><count>; 13 - verbose; 14 -
normal; 15 - silent; 16 - help; 17 - max <file>
> 8
Current multiset:
foo 9
bar 8
```

```
baz 6
boz 1
buz 4
0 – exit; 1 – input <file>; 2 – union <file>; 3 – subtract <file>;
4 – difference <file>; 5 – intersect <file>; 6 – reset current set
7 – write <file>; 8 – print; 9 – find <item>; 10 – insert
<item><count>;
11 – delete <item>; 12 – reduce <item><count>; 13 – verbose; 14 –
normal; 15 – silent; 16 – help; 17 – max <file>
> 2 a
a union completed
0 – exit; 1 – input <file>; 2 – union <file>; 3 – subtract <file>;
4 – difference <file>; 5 – intersect <file>; 6 – reset current set
7 – write <file>; 8 – print; 9 – find <item>; 10 – insert
<item><count>;
11 – delete <item>; 12 – reduce <item><count>; 13 – verbose; 14 –
normal; 15 – silent; 16 – help; 17 – max <file>
> 1 a
New Set Loaded
0 – exit; 1 – input <file>; 2 – union <file>; 3 – subtract <file>;
4 – difference <file>; 5 – intersect <file>; 6 – reset current set
7 – write <file>; 8 – print; 9 – find <item>; 10 – insert
<item><count>;
11 – delete <item>; 12 – reduce <item><count>; 13 – verbose; 14 –
normal; 15 – silent; 16 – help; 17 – max <file>
> 14
> 8
Current multiset:
foo 5
bar 4
baz 2
boz 1
> 3 b
b subtraction completed
> 8
Current multiset:
foo 1
boz 1
> 4 b
b difference completed
> 8
Current multiset:
foo 3
boz 1
bar 4
baz 4
buz 4
> 1 a
New Set Loaded
> 8
Current multiset:
foo 5
bar 4
baz 2
boz 1
```

```
> 4 b a
b difference completed
> Not a valid number. Please reenter a number: 8
Current multiset:
foo 1
baz 2
boz 1
buz 4
> 1 a
New Set Loaded
> 4 b a
b difference completed
> Not a valid number. Please reenter a number: ^C
sun01:14% ./proj2
> 1 b
New Set Loaded
> 8
Current multiset:
foo 4
bar 4
baz 4
buz 4
> 9 foo
Item foo found with count 4
> 9 foo 4
Invalid input!
please reenter the <item name> (Enter STOP to quit):
STOP
> 10 foo
Invalid input!
<foo>'s <count> is missing
Please reenter your < item name> and <count>:
foo 3
Item foo inserted with count 7
> 10 bar -2
Item bar inserted with count 2
> 9 booz
Item booz not in multiSet
> 8
Current multiset:
foo 7
bar 2
baz 4
buz 4
> 11 buz 4
Invalid input!
please reenter the <item name> (Enter STOP to quit):
buz
Item buz deleted
> 12 bar 5
Item bar removed
> 12 foo 5
Item foo count reduced to 2
> 8
Current multiset:
```

```
foo 2
baz 4
> 12 baz 4
Item baz removed
> 16
================================================================
Usage: proj2 [-s][-v][-h][-f] <filename>]
        -s: silent mode
        -v: verbose mode
        -h: print this help
        -f <filename>: read <filename> into the current set
================================================================
The numbered commands are as follows:
0. exit
1. input file <filename>: open and read a list from a file to the
current multiset
2. union file <filename>: open and union a multiset from a file with
the current multiset
3. subtract file <filename>: open and subtract multiset from a file
from the current multiset
4. difference file <filename>: open and find the difference between a
multiset from a file and the current multiset
5. intersect file <filename>: open and find the intersection between a
multiset from a file and the current multiset
6. reset current set to the empty multiset
7. output file <filename>: open and write the current multiset to a
file
8. print current multiset to the console
9. find <item name>: test if <item name> is in the current multiset
10. insert <item name><count>: add <item name> to the current multiset
with number <count> if <item name> is not in the current multiset, or
increase <item name>'s number by <count> if it is
11. delete <item name>: remove <item name> from the current multset if
it is in it
12. reduce <item name><count>: reduce the number of <item name> by
<count>
13. verbose output
14. normal output
15. silent output
16. help
17. max file <filename>:
================================================================

> 0

sun01:3% ./proj2 -v -f a
0 - exit; 1 - input <file>; 2 - union <file>; 3 - subtract <file>;
4 - difference <file>; 5 - intersect <file>; 6 - reset current set
7 - write <file>; 8 - print; 9 - find <item>; 10 - insert
<item><count>;
11 - delete <item>; 12 - reduce <item><count>; 13 - verbose; 14 -
normal; 15 - silent; 16 - help; 17 - max <file>
New Set Loaded
> 8
Current multiset:
```

```
foo 5
bar 4
baz 2
boz 1
0 – exit; 1 – input <file>; 2 – union <file>; 3 – subtract <file>;
4 – difference <file>; 5 – intersect <file>; 6 – reset current set
7 – write <file>; 8 – print; 9 – find <item>; 10 – insert
<item><count>;
11 – delete <item>; 12 – reduce <item><count>; 13 – verbose; 14 –
normal; 15 – silent; 16 – help; 17 – max <file>
> 14
> 6
Reset completed
> 1 a
New Set Loaded
> 17 b
Max completed
> 8
Current multiset:
foo 5
bar 4
baz 4
boz 1
buz 4
> 0
```