Programmer’s Guide and Report for

Project 4- Revised and augmented sets, multisets

**Source:**

Header files:

*Proj4.h*

Application files:

*testP4main1.cpp*

*testP4main2.cpp*

*testP4main3.cpp*

Makefiles:

*makefile1 - Set*

*makefile2 - MultiSet*

*makefile3 - class hierarchy test*

**Authors’ names:** Cory Turco, Wenlan Tian

**Authors’ contributions:** Cory Turco wrote almost all the functions, and Wenlan did the modifications, testing, wrote the documents.

**Date last modified:** 08-05-2014

**Problem Statement**

This program takes the set and multiset specifications and overloads a number of operators: +/+=(union), -/-=(subtraction), =(deep copy), \*/\*=(intersect), ^/^=(difference), ==(equality), <(proper subset), <=(subset), <<(stream out) and >>(stream in).

Templates are used to allow a set or multiset holds items of any type. Multiset has an integer associated with each item that indicates the number of that item in the multiset.

The base MySet class and a derived MyMultiset class from mySet were implemented.

**Organization of Code**

This program defined two template classes: MySet and MyMultiSet.

MySet is the base class. Set is a vector of <T>, which could be any type (int, double, string, characters, etc.).

MyMultiSet is a derived class from MySet class. Multiset is an extension of set, with overriding functions where needed. Multiset holds the items as set, but also have an integer associated with each item indicating the number of that item. A vector<int> count was used to store these numbers.

In MySet class, the key word ***virtual*** was used for the functions, which will be overridden in MyMultiSet.

**Functions, Methods, Procedures**

A total of 14 operators were overloaded for Set and MultiSet respectively in this program.

operator+()- union a set/multiset from a file with the current set/multiset, but not return the current set/multiset

operator-()- subtract set/multiset from a file from the current set/multiset, but not return the current set/multiset

operator^()- find the difference between a set/multiset from a file and the current set/multiset, but not return the current set/multiset

operator\*()- find the intersection between a set/multiset from a file and the current set/multiset, but not return the current set/multiset

operator+=()- union a set/multiset from a file with the current set/multiset, and return the current set/multiset

operator-=()- subtract set/multiset from a file from the current set/multiset, and return the current set/multiset

operator^=()- find the difference between a set/multiset from a file and the current set/multiset, and return the current set/multiset

operator\*=()- find the intersection between a set/multiset from a file and the current set/multiset, and return the current set/multiset

operator=()- deep copy a set/multiset from a file to the current set/multiset

operator==()- test if a set/multiset from a file is the same the current set/multiset

operator<()- test if a set/multiset from a file is a proper subset of the current set/multiset

operator<=()- test if a set/multiset from a file is a subset of the current set/multiset

operator<<()- stream out the set/multiset

operator>>()- stream in to a set/multiset

additional function:

compare() – test whether the set/multiset from a file is identical the current set/multiset, or a proper subset/multiset of the current set/multiset; test whether the current set/multiset is a proper subset of the set/multiset from a file; the set/multiset from a file is not comparable with the current set/multiset.

Helper functions:

Set:

clear()- reset current set to the empty set

find(T)- test if <item name> is in the current set

addElt(T)-add <item name> to the current set if it is not already in it

deleteElt(T)-remove <item name> from the current set if it is in it

get\_name(int)- get the name when giving the location

size() – return the size of the set

get\_location – get the location index of a item

MultiSet:

clear()- reset current multiset to the empty multiset

addElt(T,int)-add <item name> and <count> to the current multiset if it is not already in it

deleteElt(T)-remove <item name> and <count> from the current multiset if it is in it

reduce(T, int)- reduce the <count> of <item name>

item\_count(T) – set the count of an item

get\_count(int) – get the count of a item

**Known Bugs**

1. After a certain type is selected, if the input item type is different from the selected type, the program cannot recognize this error.

**Testing**

<1> Set testing was done by testP4main1.cpp and four files: f1, f2, f3, f4.

[f1]

banana

apple

grapefruit

orange

pear

grape

lime

[f2]

banana

mango

pear

lychee

avocado

lime

apple

guava

[f3]

lime

banana

apple

grapefruit

guava

orange

pear

grape

mango

[f4]

orange

banana

apple

grapefruit

pear

grape

lime

UFs-MacBook-Pro:Proj4 tianwenlan$ ./Proj4\_Set

0-quit, 1-ints, 2-chars, 3-string, 4-doubles, 5-help

type > 3

Strings it is!

======================================================================

The numbered set commands are as follows:

0. exit

1. input file <filename>: open and read a set from a file to the current list

2. union file <filename>: open and union a set from a file with the current set

3. subtract file <filename>: open and subtract set from a file from the current set

4. difference file <filename>: open and find the difference between a set from a file and the current set

5. intersect file <filename>: open and find the intersection between a set from a file and the current set

6. reset current set to the empty set

7. output file <filename>: open and write the current set to a file

8. print current set to the console

9. find <item name>: test if <item name> is in the current set

10. insert <item name>: add <item name> to the current set if it is not already in it

11. delete <item name>: remove <item name> from the current set if it is in it

13. verbose output

14. normal output

15. silent output

16. help

17. compare <file>: read a set from a file, print

-1 if current set is properly contained in the new set

0 if current set is identical to the new set

1 if current set is properly contains the new set

2 if current set and the new set are not comparable

18. equal <file>: read a set from a file, print 1 if current set is equal to the new set, else 0

19. less than <file>: read a set from a file, print 1 if current set is properly contained in the new set, else 0

======================================================================

String Sets > 1 f1

New set loaded

String Sets > 8

banana

apple

grapefruit

orange

pear

grape

lime

String Sets > 2 f2

String Sets > 8

banana

apple

grapefruit

orange

pear

grape

lime

mango

lychee

avocado

guava

String Sets > 3 f2

String Sets > 8

grapefruit

orange

grape

String Sets > 6

Reset completed

String Sets > 8

String Sets > 1 f1

New set loaded

String Sets > 4 f2

String Sets > 8

grapefruit

orange

grape

mango

lychee

avocado

guava

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 5 f2

String Sets > 8

banana

apple

pear

lime

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 17 f4

0

String Sets > 17 f3

-1

String Sets > 17 f2

2

String Sets > 6

Reset completed

String Sets > 1 f3

New set loaded

String Sets > 17 f1

1

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 18 f4

1

String Sets > 18 f2

0

String Sets > 19 f4

0

String Sets > 19 f3

1

String Sets > 20 f2

String Sets > 8

banana

apple

grapefruit

orange

pear

grape

lime

mango

lychee

avocado

guava

String Sets > 21 f2

String Sets > 8

grapefruit

orange

grape

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 22 f2

String Sets > 8

grapefruit

orange

grape

mango

lychee

avocado

guava

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 23 f2

String Sets > 8

banana

apple

pear

lime

String Sets > 6

Reset completed

String Sets > 1 f1

New set loaded

String Sets > 24 f4

1

String Sets > 24 f3

1

String Sets > 24 f2

0

String Sets > 0

type > 0

bye bye

<2> MultiSet testing was done by testP4main2.cpp and five files: a, b, c, d, e

[a]

foo 5

bar 4

baz 2

boz 1

[b]

foo 4

bar 4

baz 4

buz 4

[c]

foo 5

baz 2

boz 1

bar 4

buz 6

[d]

foo 3

bar 4

baz 2

boz 1

[e]

foo 3

bar 45

foo bar

bike car 33

banana 18 19

orange -3

apple

grape 14

UFs-MacBook-Pro:Proj4 tianwenlan$ ./Proj4\_MultiSet

0-quit, 1-ints, 2-chars, 3-string, 4-doubles, 5-help

type > 3

Strings it is!

======================================================================

The numbered multiset commands are as follows:

0. exit

1. input file <filename>: open and read a multiset from a file to the current list

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 multiset 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>: add <item name> to the current multiset if it is not already in it

\*11. delete <item name>: remove <item name> from the current multiset if it is in it

13. verbose output

14. normal output

15. silent output

16. help

\*17. compare <file>: read a multiset from a file, print

-1 if current multiset is properly contained in the new multiset

0 if current multiset is identical to the new multiset

1 if current multiset is properly contains the new multiset

2 if current multiset and the new multiset are not comparable

18. equal <file>: read a multiset from a file, print 1 if current multiset is equal to the new multiset, else 0

19. less than <file>: read a multiset from a file, print 1 if current multiset is properly contained in the new multiset, else 0

\* - these commands may or may not be implemented

======================================================================

String Multisets > 1 a

New multiset loaded

String Multisets > 8

foo 5

bar 4

baz 2

boz 1

String Multisets > 2 b

String Multisets > 8

foo 9

bar 8

baz 6

boz 1

buz 4

String Multisets > 3 b

String Multisets > 8

foo 5

bar 4

baz 2

boz 1

String Multisets > 4 b

String Multisets > 8

foo 1

baz 2

boz 1

buz 4

String Multisets > 6

String Multisets > 8

String Multisets > 1 a

New multiset loaded

String Multisets > 5 b

String Multisets > 8

foo 4

bar 4

baz 2

String Multisets > 6

String Multisets > 1 a

New multiset loaded

String Multisets > 17 c

-1

String Multisets > 17 d

2

String Multisets > 17 a

0

String Multisets > 6

String Multisets > 1 c

New multiset loaded

String Multisets > 17 a

1

String Multisets > 6

String Multisets > 1 a

New multiset loaded

String Multisets > 18 a

1

String Multisets > 19 d

0

String Multisets > 19 c

1

String Multisets > 6

String Multisets > 1 a

New multiset loaded

String Multisets > 20 b

String Multisets > 8

foo 9

bar 8

baz 6

boz 1

buz 4

String Multisets > 21 b

String Multisets > 8

foo 5

bar 4

baz 2

boz 1

String Multisets > 23 b

String Multisets > 8

foo 4

bar 4

baz 2

String Multisets > 6

String Multisets > 1 a

New multiset loaded

String Multisets > 24 a

1

String Multisets > 24 b

0

String Multisets > 6

String Multisets > 1 e

<foo>'s <count> is missing

<bike>'s <count> is missing

This line is empty

line malformed: banana 18 19

[orange] has invalid count number! <Count> should be larger than 0

[apple] is not in correct format on user input line

New multiset loaded

String Multisets > 8

foo 3

bar 45

banana 18

grape 14

String Multisets > 0

type > 0

bye bye