CS 2123-001 Data Structures

Instructor Dr. Turgay Korkmaz

Homework 4
Due date: check BB Learn
!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

(Abstract Data Type - Library)
Background [see http://en.wikipedia.org/wiki/Set_(mathematics)]:
In mathematics, a **set** is a collection of distinct elements (say integer numbers). For example, A={2, 5, 7} and B={3, 5, 8, 10} are two different sets. There are several basic operations for constructing new sets from given two sets, but let's just consider three basic ones:

C = union(A,B); ---> C = $A \cup B$={2, 3, 5, 7, 8, 10}
C = intersection(A,B); ---> C = $A \cap B$={5}
C = difference(A,B); ---> C = A-B = $A \setminus B$ ={2, 7} also aka. complement

What to do:
You are asked to develop a **setADT** as a library (using two different representations to store the elements in the set: **array** and **link list**). So you will have `setArrayImp.c` and `setLinkedListImp.c`
Then you will implement a driver program that gets two sets from the user and one of the above operation as a command to apply to create the new set. It prints the resulting set...

Developing setADT as a library:

First, create an interface file set.h which contains boiler plate and the followings:

```
typedef int setElementT;
typedef struct setCDT *setADT;

setADT setNew();            /* create a new empty set */
void setFree(setADT S);     /* free the space allocated for the set S  */

int setInsertElementSorted(setADT S, setElementT E);
```

/* if not successful, return 0; otherwise, return the number of elements in the set (including the element just inserted). If the same element is given again, simply ignore it because a set contains distinct elements. Also note that the elements might be given in any order, but your function should always keep the set in a sorted manner after each insertion */

```
setADT setUnion(setADT A, setADT B);
                                    /* returns a new set containing A ∪ B        */

setADT setIntersection(setADT A, setADT B);
                                    /* returns a new set containing A ∩ B        */
```

```
setADT setDifference(setADT A, setADT B);
                                    /* returns a new set containing A \ B          */

int setCardinality(setADT S);          /* return the number of elements in S */

void setPrint(setADT S, char *name); /* print elements of S, A = {2, 5, 7}.
```
name is the name of the set that needs to be printed before printing the elements */

Second, you need to implement the exported functions and decide how to store/represent the set elements (i.e., give the details of `struct setCDT)`. As we did in the implementation of `queue.h`, you are be asked to have <u>two different</u> representation and implementations: `setArrayImp.c` and `setLinkedListImp.c`

- (40%) First implement set library as setArrayImp.c which uses a constant size array to store set elements (suppose max set size is 100). [*hint: see ch2 slides 73-79*]
- (40%) Second implement this library as setLinkedListImp.c which uses a dynamic single linked list to store set elements.

Developing a driver/application using setADT

Implement a driver.c program and compile it with two different imp of set library (20%)

Here is the basic steps for your driver application:

1. Create two sets called A and B.
2. Ask user to enter positive integers for set A (end input when user enters -1)
3. Ask user to enter positive integers for set B (end input when user enters -1)
4. In a loop
    - 4.1. Ask user to enter a command:
    - 4.2 If Q is entered, quit from this loop.
    - 4.3 If U, I, or D is entered, compute set C as union, intersection, or difference.
        - setPrint(A, "A"); setPrint(B, "B"); setPrint(C, "C");
        - print the number of elements in C
        - setFree(C);
5. free A and B

Create a Makefile to compile driver.c with setArrayImp.c as well as setListImp.c

/* Don't forget to include comments about the problem, yourself and each major step in your program! */

What to return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

DO ALL YOUR WORK UNDER abc123-hw04 folder using your own abc123

1. To easily compile the set library and driver.c program with setArrayImp.c and setLinkedListImp.c, you must have a `Makefile` and use "`make`" to compile your code.

make array: should compile it with setArrayImp.c;

make list: should compile it with setLinkedListImp.c

2. After compiling, run both versions of your program a few times with different input values, then save the output (using script) into `out4.txt` file.
3. Remove executables from **abc123-hw04** folder and then Zip that folder
4. Then go to BB Learn, and submit your **abc123-hw04.zip** as an attachment before the deadline. Make sure your zip file contains all your files!

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as zip file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.

Here is a Makefile for you:

```
all: array list

setArrayImp.o: setArrayImp.c set.h
     gcc -c setArrayImp.c

setLinkedList.o: setLinkedListImp.c set.h
     gcc -c setLinkedList.c

driver.o: driver.c set.h
     gcc -c driver.c

array: driver.o setArrayImp.o
     gcc driver.o setArrayImp.o  -o driver-array

list: driver.o setLinkedListImp.o
     gcc driver.o setLinkedListImp.o -o driver-list
```