CS xyz3-001 Foundations of Programming and Data Structures

Instructor Dr. Turgay Korkmaz

Homework 10
Due date: check BB
!!!! 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

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

1. Create a directory `abc123-hw10`, using your own abc123. Do all your work under that directory.

2. To easily compile the set library with two different implementation and driver program, you must have a Makefile and use “make” to compile your code. (see the sample Makefile at the end of this document)

3. Follow the problem solving methodology, and solve the problem(s). Then convert your solution(s) to a C programs. You can name your interface file as `set.h` and two different implementations of set ADT as `setArrayImp.c` and `setLinkedListImp.c`. Then you can name your driver program here as `hw10.c`

```
/*
 * Don't forget to include comments about the
 * problem, yourself and each major step in your
 * program! so that we can understand your
 * solution(s).
 */
```

4. Compile your set ADT and driver program using Makefile. Then run it and copy/paste the results in an output file, which you can name as hw10-out.txt. Also make sure you get hw10-valgrind.txt, as described in previous assignments.

5. Zip the whole directory `abc123-hw10 as abc123-hw10.zip`
6. Go to BB Learn (http://learn.utsa.edu/) , login using your abc123
7. Submit your `abc123-hw10.zip` for hw10 under Assignments

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

1) All assignments must be submitted as either a zip or tar archive 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 sample Makefile for you:

```
all: array list

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

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