

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_\(mathematics\)](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:

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

What to do:

You are asked to develop a **set** library (using two different representations: **array** and **link list**) and then implement a driver program that gets two sets and one of the above operation as a command to apply, then it prints the resulting set...

Developing set library:

For interface, 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). Also note that the elements might be given in different
orders, 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  $A \cup B$  */

setADT setIntersection(setADT A, setADT B);
/* returns a new set containing  $A \cap B$  */

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

```
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} */
```

For implementation, you will be asked to have two different implementations:

setArrayImp.c and setLinkedListImp.c

- (40pt) 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]
- (40pt) Second implement this library as setLinkedListImp.c which uses a dynamic single linked list to store set elements.

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

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(A, "B"); setPrint(C, "C");
 print the number of elements in C
 setFree(C);
5. free A and B

Compile/execute 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 !!!

1. Create a directory, say LASTNAME_hw4, and do all your work under that directory.
2. 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
3. After compiling, run both versions of your program a few times with different input values, then save the output (using script) into output.txt file.
4. you will have around 6-7 files in your LASTNAME_hw4 directory.
5. Go to parent directory of LASTNAME_hw4, and use
 > tar -cf LASTNAME_hw4.tar LASTNAME_hw4

This will create a new file called LASTNAME_hw4.tar and it contains all of your files. So just submit this .tar file.

6. Go to BB Learn, and just submit LASTNAME_hw4.tar as **attachment** before the deadline. DO NOT submit other .h or .c files individually.

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. _____