Birla Institute of Technology & Science, Pilani
2nd Semester 2016-17 - CS F211 – Data Structures and Algorithms

---

Lab 2 - 24th Jan 2017
Topics – Abstract Data Types and Performance Measurements

1. Implementing Abstract Data Types in C

An Abstract Data Type is realized by separating the user concerns (types and interface) from provider concerns (implementation). In C,

- a header file (with suffix ".h") is used to define the data type and declare the functions / procedures operating on it
- a code file (with suffix ".c") is used to define the implementation (I.e. define the functions / procedures).

For instance, consider the ADT CircularStack, where the Stack is of bounded size, and a push operation overwrites the oldest value when the Stack is full.

We use cstack.h and cstackImpl.c to realize this ADT:

```
/* cstack.h */
typedef … *CircularStack;
Element top(CircularStack);
…

/* cstackImpl.c */
Element top(CircularStack) {  retun … ; }

…
```

Then, a (test) program that uses CircularStack, say History, can be implemented as a code file, say hist.c:

```
#include "cstack.h"
int main(int argc, char **argv)
{
  CircularStack cs;
  …
   e = top(cs);
   …
  return 0;
}
```

Exercise: Use this approach to realize ADT CircularStack using a circular linked list of a given length L, as the representation. Test your implementation by building up a history larger than the L, print the history, add L+k more elements, and print the history.

2. Measuring (Dynamic) Memory Used and Time Taken

a. **Measuring Memory:** The amount of dynamic memory allocated can be kept track by programming. Define the following:

- a global array, heapAllocs, records <pointer, size>, keeping track of (heap) space allocated for each pointer
- a global variable, curHeapSize, keeping track of total (heap) space currently allocated
- a global variable, maxHeapSize, keeping track of the maximum total (heap) space allocated in this run of this program.

Define procedures *myAlloc* and *myFree* such that:

myAlloc calls *malloc* but also adds an element to heapAllocs, increments curHeapSize by the amount requested for allocation, and updates maxHeapSize if curHeapSize exceeds the previous maximum.

myFree calls *free* but also decrements curHeapSize, and deletes the record corresponding to the freed pointer from heapAllocs

In the end, you shall need to print heap memory left (i.e. not freed) and maximum heap memory allocated during the entire execution. You shall need to print the values in bytes.

b. The amount of running time can be measured using the *time* function. Use #include <time.h> this function. To measure the execution time of a particular function, you need to measure the clock time using *time* function before and after the function call and then take difference of the two. Refer to man pages on how to use this function.

Exercise:

(a) Given a file containing a list of integers in increasing order, create a sorted list, Ls, implemented as a dynamically allocated array. Measure the time taken for several random insertions into Ls. Also, measure the total heap space allocated.
(b) Repeat the exercise where the sorted list is implemented as a linked list.