Programming Guidelines

Some programming guidelines for the completion of the programming requirements of this course are listed below. We expect you to submit a SINGLE FILE that contains: (a) your C++ code, and (b) experimental results included in the form of C/C++ comments. Do not forget to identify yourself in the first line of the code file. look like

```
/* Alex. Gerbessiotis  HW1 Programming **** This is a C-style comment */
/* Experimental results are included here. This line
   may extend on multiple
   lines such as this
   example
 */

 // Code follows; Try to avoid C++ oriented comment lines.
```

The name of the file is `hwXsYZW.c` file, `X` is the homework number and `YZW` are the last three digits of your ID number.

Make sure that the entry points of your functions adhere to the format outlined in the programming problem description.

Submit the `hwXsYZW.c` file by e-mail to the course account address `alg435@cs.njit.edu` as an ASCII text file including the label `hwXsYZW` of the file in the Subject line of the e-mail. *Make sure* that your mailer DOES NOT send it as an attachment.

For example a subject line can look like the following one.

```
Subject:  hw1s234
```

for the Homework 1 related submission of a students with id ending in 234.

The `hwXsYZW` file MUST contain in the form of C comments a description of the various functions defined (i.e. for each function used explain what it does and what its various arguments denote). The code you submit must contain the function(s) to be implemented. If you also require some extra functions in your code their names ought to begin with the `hwXsYZW` string as well. Note that you need to do your testing separately. It is imperative therefore that your submitted file DOES NOT CONTAIN a `main()` function.

It is your responsibility that your code in ANSI C++ or ANSI C compatible and compilable. We can only tell you that testing will be done on a SUN Workstation or Linux-based PC using gcc.

No partial credit will be given to submitted code that does not satisfy the previous guidelines. No partial credit will be given to submitted code that it is not compilable. If your code fails one of the guidelines, we will not request retransmission.

No partial credit will be given for code that does not fully list its bugs. The grader will decide testing instance(s) and grade your submission based on whether it passes successfully or not these testing instances.

C versus C++ Some of you may complain that malloc/free are C and not C++. They are there in any C++ compiler. If you don't know how to use them, then you will learn a little more about C++ that you didn't know. If you already know how to use them, then you can practice more with their usage. On an AFS account `man malloc` may provide some extra documentation. We intentionally use malloc/free as opposed to new/delete to make the assignments more interesting, incompatible with published web code.

Visual C++ vs AFS Although testing will occur on an AFS machine, it is not required to write your code on AFS. If you follow our guidelines then you can use the Visual C++ environment to write your ANSI C++ or ANSI C compatile code and we can compile it separately.

1 Supplied material

In the course Web-page you may find a file named `testing.tar`. It is a UNIX tar file that can be expanded by typing say on a Unix command prompt `%`

```
% tar xvf testing.tar
```

Winzip on a PC also knows how to deal with tar files.

The three files of this .tar file, `Makefile`, `bubble.c` and `sortg.c` are also available individually. The source code of the second file also appears in this document. `Makefile` shows how to compile and link these files under Unix. An example of a sorting algorithm (bubble sort) is provided in `bubble.c`. This example also shows how the arguments that are common to all functions of this assignment operate. File `sortg.c` shows how to call the function in `bubble.c` and how to define a specific `compare` function (See discussion in part A of the sorting oriented programming assignments when they become available). Although in part B of the programming assignments you are going to test your algorithms on input arrays of integers, the testing functions of the grader may test your code on sequences whose elements are `double`, strings, etc. *It is imperative that you not make any assumption on the date type of the input sequence.*

Makefile. If you don't know how to use a Makefile you can import `sortg.c` into your project and let do as you please with it without the Makefile. If you use AFS it will be of assistance to use the Makefile. If you don't know about make, type in AFS `man make` and you will find out more. Note tha having a makefile and a Makefile at the same directory is dangerous and/or confusing. Visual C++ has a make program called `nmake`. You can compile code in Visual C++ through a makefile as well; don't ask us how, however. Read the help topics in Visual C++ or look up on the Web. GNU Make is also freely available under the GNU public license and already installed on AFS. You can get extensive information through the Web as well.

Programming Assistance. We respond only to questions related to the programming assignments. If you don't know how to use Visual C++, this course will not teach how to program. CIS 113 and CIS 114 and the new CIS 288 might. You are supposed to know how to use some environment to program in C++. We don't care whethere you program on a PC with Visual C++, on a Linux PC using gcc, or an AFS machine using whatever compiler it has. The end results is that your code should be compilable in our ANSI C++ or ANSI C compatible testing platform.

Segmentation Faults/Core dumping If you get a related message, then your program has a bug that is pointer related. Debug it! Don't ask us how. Every CS major should know how.

Can Java be used instead? No.