

CGS 3460 Computer Programming Using C, Spring 2008

Homework 5

Due Monday, March 31 2008, before 11:59:59pm

Notes

- All submissions must be done electronically via the Courseworx system linked from the website.
- Create a separate C source file for each problem. Problem 1 must be in `p1.c` and Problem 2 in `p2.c`.
- Use `tar` to combine all the files into the file `H5.tar`. To do this, type `tar -cvf H5.tar p1.c p2.c` on the command line. You must only upload `H5.tar` to Courseworx.
- Once you've uploaded the file, download the file and untar it using the command `tar -xvf <filename>`. Display each extracted file to verify your programs are intact.
- Submit C source files *only* (files with `.c` extension). We will compile and run them.
- The first three lines of each C source file must contain your Full Name, UFID and Gatorlink ID as comments.
- Make sure that your code compiles and runs correctly on one of the following CISE machines: `sand.cise.ufl.edu`, `rain.cise.ufl.edu`, `shine.cise.ufl.edu`, `bay.cise.ufl.edu`.
- When obtaining input, be sure to prompt the user appropriately.

1. Design a structure to store a number, which is either an integer or a floating point number. The structure must contain a union which has a `float` and an `int` variable. Additionally the structure uses an enumeration to keep track of which one of them is currently being stored. Create functions `add_number`, `subtract_number` and `multiply_number` that given two numbers return the number obtained by performing the appropriate operation. You will have to keep track of types, so that the operations give correct results. Demonstrate the use of these functions in `main()`. Ask the user to enter two numbers, obtain each number as a string, and figure out if it is an integer or a floating point number and then store it in the structure variable. Assume that the user will enter either an integer or a floating point number (no error checking needed).

2. Design a structure to store a complex number. The real and imaginary parts of the complex number must be numbers (as created in the previous problem). Create two functions called `add_complex` and `multiply_complex` which each take two instances of the structure and return their sum and product respectively. Demonstrate the use of these two functions in `main()` by getting two complex numbers from the user and displaying their sum and product. The user will enter each complex number of the form `a+ib`, where `a` and `b` are numbers and $i^2 = -1$. Obtain the input (each complex number) from the user as a string, and write a function `string_to_complex` that given such a string returns an equivalent instance of the structure you define above. You may assume that the strings are of the form `a+ib` (no error checking needed). Here is a sample interaction with the program (User input is in bold):

```
Enter first complex number: 3.25+i0.77
Enter second complex number: 5+i2.5
Sum: 8.25+i3.27
Product: 14.325+i11.975
```