# **­CS241 Homework 1, Winter 19**

Data types and functions

**Due:** Wednesday, Feb 13, 2019

***Please answer all of the questions and submit your work through Blackboard. I can also print out a hard copy and turn in your hard copy in class.***

1. (10 points) Give at least 5 keywords that are explicitly reserved in standard ANSI C and briefly explain their use.

Break – terminates a loop

if – function that evaluates an expression true or false

else – comes after an if statement to execute when the expression is false

unsigned – alters a base data type(int), no negatives, bigger positive range

goto – unconditional jump to a place in the code

1. (10 points) Name 5 of the six types of tokens defined in ANSI C and give two examples for each of them.

Keywords – int, while

identifiers – main, total

constants – 10,20

special symbols - () {}

operators - + -

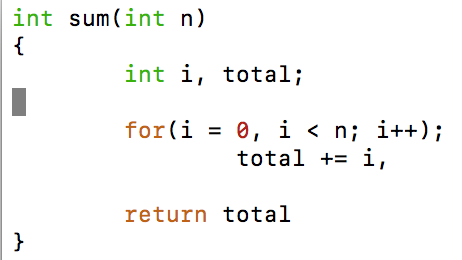
1. (10 points) Which of the following are not identifiers in ANSI C and why? Points will be taken off if your answer is wrong.
   1. 3id / not – starts with a number
   2. \_good
   3. cis.gvsu / not – contains “.”
   4. x2go
   5. YouAreGood
   6. starwall\* / not – contains “\*”
   7. int not – / int is a keyword in ansi c
   8. xYshouldI
   9. me\_to-1 / not – contains “-”
   10. gogo00
   11. float0int
   12. \_\_yes
   13. ThisOneIsNot
   14. badOne
   15. shouldOK
2. (10 points) Please complete the following calculations in an ANSI C environment.
   1. 7 / 2.0 = 3.5
   2. 2 + 5 \* 2 = 12
   3. (5 + 2) \* 2 = 14
   4. 5 \* 2 / 3 = 3
   5. 5 \* (2 / 3) = 0
3. (10 points) We know that different number of bytes are used to represent different data types in C. For example, one byte is used for **char**, two bytes are used for **short**, four bytes are used for **int**, and so on.
   1. What are the tradeoffs of using different data types in your program?

Using smaller data types take less memory and in theory, speed up the program, but limit the amount of data. Bigger data types require more memory and may slow down the program.

* 1. How can you ensure that the value being assigned to a variable is within the range that the data type of the variable can hold?

Use sizeof()

1. (10 points) Find the errors in the following code and briefly explain them. This function is supposed to compute the sum of *1 – n*. Some are grammar errors. Some are logical errors.



total is not initialized

, instead of ; after total+=i

missing ; after return total

; after the for loop line, should be blank or {}

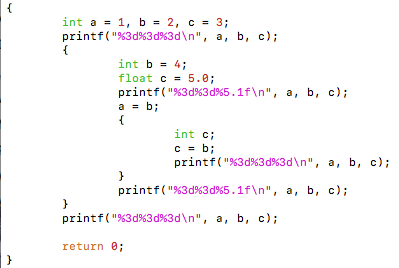
should be i <= n or i < n+1 to include n in the total

1. (10 points) What are the differences between a C function and a Java method?

Functions do not have any reference variable, can be defined anywhere, no overloading

methods are called using reference variables, must be defined in a class, overloading possible

1. (15 points) What will get printed when the following piece of code is executed?



1 2 3

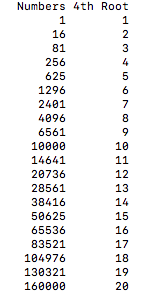
1 4 5.0

4 4 4

4 4 5.0

4 2 3

1. (10 points) Use the library function **sqrt()** to write a function that returns the fourth root of its **int** argument ***k***. The value returned should be a **double**. Use your function to generate a table that looks like the one shown below. The first column is the input number; while the second column is the fourth root of that number. Note that your program must be implemented as specified. Points may be taken off otherwise. Attach your code for the answer of this question. **Hint:** Use type casting when needed.



#include <stdio.h>

#include <math.h>

double fourthRoot(int k);

int main(){

double result;

printf("Numbers 4th Root\n");

for(int i = 1; i <= 20; i++){

result = fourthRoot(pow(i, 4));

printf("%d %d\n", (int)pow(i, 4), (int)result);

}

return 0;

}

double fourthRoot(int k){

int i;

double result = (double)k;

for(i=0;i<2;i++){

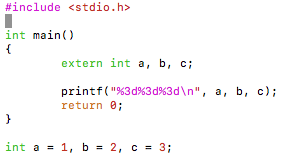
result = sqrt(result);

}

return result;

}

1. (15 points) Use the program below to answer the following questions.



* 1. What will get printed? Explain your answer.

1 2 3, extern defines a, b, and c as global variables and can be declared again throughout the program, but only initialized once.

* 1. Change the last line of the program to

**static int a = 1, b = 2, c = 3;**

Compile the program again. What will happen? Explain your observation.

An error is thrown, the variables can’t be declared static after they were already declared non static by extern int

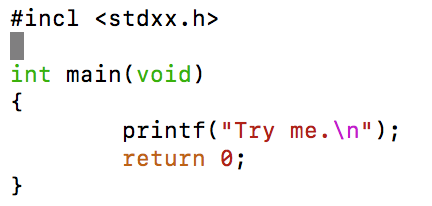
* 1. Change the first line in the **main()** function to

**int a, b, c;**

Compile the program again. What will get printed? Explain the results.

It prints 32766, 0, 0, this is because the static int a = 1 c = … is not in the scope of the main function and it prints out uninitialized variables.

1. (10 points) When you invoke a C compiler like gcc, the system first invokes the preprocessor. In this exercise, we want to deliberately make a preprocessing error, just to see what happens. Try the following program:



* 1. Type in the above code exactly as what it is. Compile it and observe what happens. Summarize your observation.

It tries to process the #incl but doesn’t know what #incl is

* 1. What happens if you change **#incl** to **#include**?

It tries to include the header file stdxx.h, but can’t find it because it doesn’t exist.