

Structures in C

1. Allowed libraries: `stdio.h`, `string.h`, `structures.h`
2. Create a Header file, which is a text file, named ***structures.h***. Do the following in ***structures.h***.

IF YOU CREATE YOUR `structures.h` FILE INCORRECTLY, YOUR PROGRAM WILL COMPILE AND RUN ON YOUR COMPUTER, BUT IT WILL FAIL TO COMPILE WHEN YOU UPLOAD TO WEB-CAT. DO EXACTLY THE FOLLOWING IN EXACTLY THE ORDER LISTED.

- a. Use `#define` to create three constants:
 - i. ***NAME_SIZE*** and set it to 80.
 - ii. ***SSN_SIZE*** and set to 13.
 - iii. ***NUM_PEOPLE*** set it to 10.
- b. Create a typedef structure named ***person_t*** which has a field for name (use ***NAME_SIZE***) called ***name***, a field for social security number with dashes named ***ssn*** (use ***SSN_SIZE***), and an integer for year of birth named ***yearOfBirth***.

The fields MUST be named exactly as shown above otherwise the tests will fail.

Make sure to define your *typedef struct* the way I showed you in the notes. Other variations that you find on the Internet will cause issues.

- c. Add prototypes for the methods created below. A prototype is simply the function declaration without the body. You can wait till you finish the functions to add them if you wish.

Here is an **example** prototype. Any function you create in `structures .cc` that gets used in `main.cc` MUST have a prototype in `structures.h`. And since all of them get used...

```
void testBits(int start_bit, int end_bit, int bit_value);
```

IF YOUR PROGRAM SEEMS TO WORK ON YOUR MACHINE OR THE CS2450 SERVER AND FAILS WHEN YOU UPLOAD TO WEB-CAT, YOU HAVE ALMOST DEFINITELY CREATED THE `structures.h` FILE INCORRECTLY. CHECK THE ORDER OF ITEMS CREATED. CHECK SPELLINGS OF VARIABLES AND FUNCTION NAMES. CHECK RETURN TYPES AND PARMETER TYPES ON YOUR FUNCTIONS. IF YOU ARE NOT SURE ABOUT SOMETHING, ASK.

3. Create a C program file named **structures.cc** which contains the functions, with bodies, listed below. Put the assignment and your name as comments as the first two lines like this.

DO NOT PUT #DEFINES IN structures.cc.

DO NOT PUT PROTOTYPES IN structures.cc.

DO NOT PUT ANY TYPEDEFS IN structures.cc.

#DEFINES, PROTOTYPES, AND TYPEDEFS SHOULD BE IN structures.h.

- a. Include the header file include below your other included files.

```
#include <stdio.h>
#include <string.h>
#include "structures.h"
```

Note the quotes instead of angle brackets on structures.h. This tells the compiler to look in the current project directory instead of the default library directory to find structures.h.

YOU ARE ONLY ALLOWED TO INCLUDE <stdio.h>,<strings.h>, and "structures.h"

- b. Create a function named **getOnePerson** which takes a reference to a **person_t** as a parameter. This function should get a single name, ssn, and year of birth from the terminal using scanf and put that data into the **person_t** structure. Use **EXACTLY** the prompts as shown in the sample output. Notice there is ONE single space after the colon. **If the prompts are not EXACTLY as described, your program will fail the tests.**

What return type should this function have? Hint, what is this function returning?

- c. Create a function named **printOnePerson** which takes which takes a Person (by value, not by reference) as a parameter. This function should print the name, ssn, and yearOfBirth from the structure in the following format

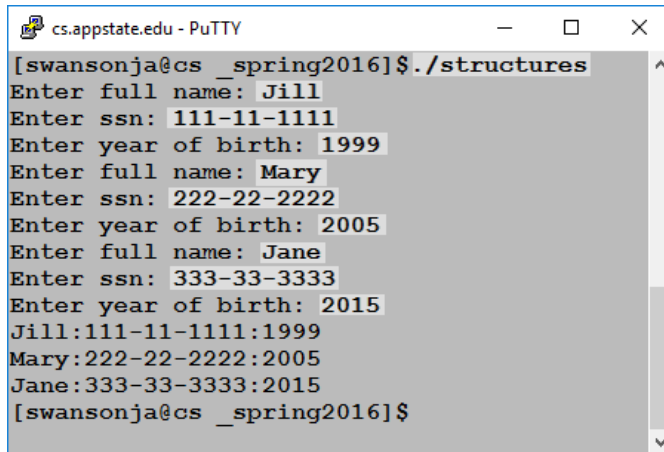
```
John_Heimerschmidt:111-11-1111:1862
```

Assuming John_Heimerschmidt, 111-11-111, and 1862 were the name, ssn, and year in the structure.

You can't have spaces in the name. There should be NO spaces before or after the colons. A newline should be printed after the year.

- d. Create a function named **getPeople** which takes an array of **person_t** structures and the number of people to get. Call **getOnePerson** for each **person_t** in the array

- e. Create a function named ***printPeople*** which takes an array of ***person_t*** structures and the number of people to print. This method should print that many Person objects from the array.
4. Copy ***main.cc*** from the AsULearn page and put it in the same folder as ***structures.cc*** and ***structures.h***.
5. Compile and make sure your output matches the output shown above for the given input. Compile command (Spaces have been exaggerated.)
`g++ -Werror -Wall -o structures main.cc structures.cc`
6. Use main to test your code. Test thoroughly before submitting. Below is a running of main.cc. The lighter background is items that the user has typed.



```
cs.appstate.edu - PuTTY
[swansonja@cs _spring2016]$ ./structures
Enter full name: Jill
Enter ssn: 111-11-1111
Enter year of birth: 1999
Enter full name: Mary
Enter ssn: 222-22-2222
Enter year of birth: 2005
Enter full name: Jane
Enter ssn: 333-33-3333
Enter year of birth: 2015
Jill:111-11-1111:1999
Mary:222-22-2222:2005
Jane:333-33-3333:2015
[swansonja@cs _spring2016]$
```

7. Submit your ***structures.cc*** on [Web-CAT](#).

Do not submit a file with a main function.

DO NOT submit structures.h.

DO NOT submit main.cc.

If Web-CAT complains that it can't find behavioral output, it means your file failed to compile on Web-CAT. Make sure your program compiles on the cs2450 server. If your program compiles on the cs2450 server but you get the above message, the most common reason for this is that you have the wrong return types on one or more of your functions or some other issue in your structures.h file. So ask if you get stuck.