CSCI 3232 Systems Software Assignment 3

Upload all your files to the dropbox in Folio before the deadline --- **11:30PM Jan 31, Tuesday, 2017.**

**Note: Make all your codes compilable and runnable under Ubuntu. Do not put your codes in Word or PDF documents. Make them into separate text files as you would compile them.**

1. (40 points) Learn the example C program in Folio consisting of three files (a.c, a.h, main.c) and complete this exercise. You need to understand function and pointer and know some sorting algorithm. Write a C program that consists of three files ***mysort.h***, ***mysort.c*** and ***myMain.c.*** Below is the ***mysort.h*** prototype

#include <stdlib.h>

#include <stdio.h>

void generateNums(int \*myarr, int len);

void sortNums(int \*myarr, int len);

* **mysort.h must contain only the prototypes listed above**
* **generateNums** function should generate **len** random integers (use *rand* function in stdlib.h) in the supplied array **myarr.** Its definition should be in mysort.c.
* **sortNums** function should sort the **len** integers at **myarr** into ascending order (you need to write your own sorting function with a sorting algorithm of your choice). Its definition should be in mysort.c.

1. In myMain.c you need to write your main function that calls the two functions in mysort.h to generate several random integers and then print out the generated unsorted random numbers in one line and the sorted numbers in another line. Your program should get **len** (which represents the number of random integers you will generate and sort) from command line argument (you can use the *atoi* function; see example code LeapYear\_cmdline.c). Upload all three files. (37 points)
2. Supply a makefile to compile your program. (3 points)

2. (40 points) You need to understand array and pointer (and pointer arithmetic) to complete this exercise. And you should be able to figure out the answers without actually compiling and running the program.

#include <stdio.h>

int main(int argc, char \*argv[])

{

char a, \*pc, c[8];

int i, \*pk, k[8];

a='N';

pc=&(c[7]);

pk=&(k[0]);

for (i=0; i<8; i++)

{

\*pc=a-(char)i;

pc--;

\*pk=i+5;

pk++;

}

return 0;

}//end of main

1. Write out the memory map for the above C code in the following table. For array variables, list the address range for the entire array. Assume the memory address starts from 100, that is, the address for **a** is 100. Suppose sizeof(**char**)=1, sizeof(**int**)=4 and the size of a memory address (a pointer) is 8 bytes. You may assume there is no gap in memory between the variables. (18 points)

|  |  |  |
| --- | --- | --- |
| Variable | Start address | End address |
| a |  |  |
| pc |  |  |
| c |  |  |
| i |  |  |
| pk |  |  |
| k |  |  |

1. Additionally, show values of the variables in the following table at the end of execution of the *for* loop before the main function returns. (For the two array variables c and k, list the contents of all array elements.) (18 points)

|  |  |
| --- | --- |
| Variable | Value |
| a |  |
| pc | Use an expression involving c to represent pc’s value |
| c |  |
| i |  |
| pk | Use an expression involving k to represent pk’s value |
| k |  |

1. How many bytes in total does the code allocate for the variables? (4 points)

3. You need to understand how to define and use a struct for this exercise. You should be able to figure out the answer without actually compiling or running the program. Which is true about the following codes? (10 points)

#include <stdio.h>

#include <string.h>

struct Books//define the struct Books

{

char title[50];

char author[50];

char subject[100];

int book\_id;

} Book1;

struct Books book;

int main(int argc, char \*argv[])

{

strcpy(book.title,"C Programming");

strcpy(book.author,"Nuha Ali");

strcpy(book.subject,"C Programming Tutorial");

book.book\_id=6495407;

strcpy(Book1.title,"Telecom Billing");

strcpy(Book1.author,"Zara Ali");

strcpy(Book1.subject,"Telecom Billing Tutorial");

Book1.book\_id=6495700;

struct Books \*struct\_pointer;

struct\_pointer = &Book1;

printf("Book1's title is %s\n",struct\_pointer->title);

printf("Book1's author is %s\n",struct\_pointer->author);

printf("Book1's subject is %s\n",struct\_pointer->subject);

printf("Book1's book\_id is %d\n",struct\_pointer->book\_id);

struct\_pointer = &book;

printf("book's title is %s\n",struct\_pointer.title);

printf("book's author is %s\n",struct\_pointer.author);

printf("book's subject is %s\n",struct\_pointer.subject);

printf("book's book\_id is %d\n",struct\_pointer.book\_id);

return 0;

}//end of main

1. Compile error b. Compiles OK, runtime error c. Compiles and runs OK d. none of the above statements is true

4. (10 points) This exercise further tests your knowledge of pointer arithmetic. Suppose we have three variables declared as

char \*pc;

int \*pi;

double \*pd;

Assume sizeof(char)=1, sizeof(int)=4, sizeof(double)=8. The values of pc, pi, pd are 240, 258, 380 respectively. What are the values of pc+1, pi+2, pd-3?