



CS-1714-0B1-Fall-2020-Computer Programming II



Exams

Review Test Submission: Exam-2 (Secure Browser)

Review Test Submission: Exam-2 (Secure Browser)

User	Rod Miles Simms
Course	CS-1714-0B1-Fall-2020-Computer Programming II
Test	Exam-2 (Secure Browser)
Started	10/30/20 9:01 AM
Submitted	10/30/20 12:25 PM
Status	Completed
Attempt	118 out of 150 points
Score	
Time Elapsed	3 hours, 24 minutes out of 4 hours
Instructions	





**Google Chrome**  
is required  
[click here](#)

Results  
Displayed

All Answers, Submitted Answers, Correct Answers, Feedback, Incorrectly  
Answered Questions

### Question 1

4 out of 4 points



What is the general form of fscanf()?

Selected Answer: ☒ fscanf (FILE \*stream, char \*format, ...)

Answers:

☐ fscanf ( char \*format, FILE \*stream, ...)

☒ fscanf (FILE \*stream, char \*format, ...)

☐ fscanf ( FILE stream, char \*format, ...)


☐ fscanf ( char format, FILE \*stream, ...)


## Question 2

0 out of 4 points



Which of the following does an object file NOT contain?

Selected Answer:  Placeholders for variables in other files

Answers:  C code

Placeholders for variables in other files

Placeholders for functions in other files


Machine code

## Question 3

4 out of 4 points




Which of the following is true for a makefile?

Selected Answer:  makefile describes to the make command that how to compile the program

Answers: makefile is a default file provided by gcc, and is not written by the user

makefile can be run directly as a normal executable file - ./makefile

 makefile describes to the make command that how to compile the program


makefile is used to perform single-step compilation

## Question 4

4 out of 4 points



What is the correct specifier in fopen() to open a binary file for writing?

Selected Answer:  "wb"

Answers: "w"

"b"

U

"wr"

✓ "wb"

### Question 5

0 out of 4 points



When parsing out the command line arguments passed into the `main()`, what is always in the first argument?

Selected



The number of arguments

Answer:

Answers:

The first argument after the executable name

A list of all the arguments that follow the name of the executable's name



The name of the executable

The number of arguments

### Question 6

4 out of 4 points



`doMagicBeta()` is a recursive function that is supposed to print out an array in reverse order.

```
1      void doMagicBeta( int *array, int number )
2      {
3          /* Line A */
4          if( number == 0 )
5          {
6              /* Line B */
7              return;
8              /* Line C */
9          }
10         else
```

```

11      {
12          /* Line D */
13          number = number - 1;
14          doMagicBeta( array, number );
15      }
16      /* Line E */
17  }


```

The initial call to this function will pass in an array of integers and the last valid index of the array (size minus one). Where do we put the following `printf( "%d\n", array[ number ] );` statement to print out the entire array in reverse order?

Selected Answer:  At `/* Line A */`

Answers:

At `/* Line E */`

 At `/* Line A */`

At `/* Line B */`

At `/* Line C */`


At `/* Line D */`

## Question 7

4 out of 4 points



What does `feof()` returns if end of file was reached?

Selected Answer:  1

Answers: 0

 1

NULL

A semi-colon

### Question 8

4 out of 4 points



To open a file in append mode what is the command that is used?

Selected Answer: ☒ "a"

Answers: ☒ "a"

☐ "w"

☐ "b"

☐ "r"

### Question 9

4 out of 4 points



What function should you call after you are done with a file that you have opened?

Selected Answer: ☒ `fclose()`

Answers: ☒ `fclose()`

☐ NULL

☐ `feof()`

☐ `fopen()`

### Question 10

4 out of 4 points



What is the value of the file pointer if the the file does not exist?

Selected Answer: ☒ NULL

Answers: ☐ EOF

☐ "New File"

☒ NULL

## Question 11

5 out of 5 points



Assume `fileIn1` is a pointer to an input file opened for reading, `fileIn2` is a pointer to another input file opened for reading. The code reads one word from both files and prints them to standard output.

```

1      char str1[ 20 ], str2[ 20 ];

2      int x, y;

3      while( ( x = fscanf( fileIn1, "%s", str1 ) ) == 1 &&
( y = fscanf( fileIn2, "%s", str2 ) ) == 1 )

4      {

5          printf( "%s\t%s\n", str1, str2 );

6      }

7      /* HERE */

```

How do you check at `/* HERE */` if `fileIn1` has more words than `fileIn2`?

Selected Answer: ☒ `if( x > y )`

Answers: ☒ `if( x > y )`

`if( x == 0 && y == 1 )`

`if( x == NULL && y != NULL )`

`if( x == y )`

`if( x != NULL && y == NULL )`

## Question 12

5 out of 5 points



Our program has the following condition at the beginning of the **main()**.

```
1    int main( int argc, char *argv[] )
2    {
3        if( argc == 4 )
4        {
5            /* do something */
6        }
7        return 0;
8    }
```

What does this code imply?

Selected

Answer: The program is expecting exactly 4 command line arguments including the name of the executable

Answers:

The program is expecting at least 4 command line arguments including the name of the executable

The program is expecting the command line argument 4

The program is expecting exactly 4 command line arguments excluding the name of the executable

The program is expecting the command line argument "4"



The program is expecting exactly 4 command line arguments including the name of the executable





You have a program that accepts two additional command line arguments: an output file name and input file name in that order.

```
1    int main( int argc, char *argv[] )
2    {
3        if( argc == 3 )
4        {
5            /* Open Files Here */
6        }
7        return 0;
8    }
```

Which of the following code correctly opens these files in `main()` at `/* Open Files Here */`?

Selected



Answer:

```
1    FILE *f1 = fopen( argv[ 1 ], "w" );
2    FILE *f2 = fopen( argv[ 2 ], "r" );
```

---

Answers:

```
1    FILE *f1 = fopen( argv[ 1 ], "w" );
2    FILE *f2 = fopen( argv[ 0 ], "r" );
```

---

```
1    FILE *f1 = fopen( argv[ 0 ], "w" );
2    FILE *f2 = fopen( argv[ 1 ], "r" );
```

---

```
1    FILE *f1 = fopen( "argv[ 2 ]", "w" );
```

```
2    FILE *f2 = fopen( "argv[ 1 ]", "r" );
```

---

```
1 FILE *f1 = fopen( "argv[ 0 ]", "w" );
2 FILE *f2 = fopen( "argv[ 1 ]", "r" );
```



```
1 FILE *f1 = fopen( argv[ 1 ], "w" );
2 FILE *f2 = fopen( argv[ 2 ], "r" );
```

### Question 14

5 out of 5 points



Given this code to read in a file named `a.txt`, what do we know about the structure of the content in `a.txt`? Assume that `fileIn` is an open pointer to a file.

```
1 char str1[ 20 ], str2[ 20 ];
2 int num;
3 while( fscanf( fileIn2, "%s %d %s", str1, &num,
  str2 ) == 3 )
4 {
5     printf( "%s %d %s\n", str1, num, str2 );
6 }
```

Selected



Answer:

The file contains a repeating pattern of string, int, string separated by a space

Answers:

The file contains a repeating pattern of string, int, string, int separated by a space

The file contains a repeating pattern of string, int, string separated by a comma

separated by a comma

The file contains alternating rows of int's and strings

The file contains alternating rows of string's and int's



The file contains a repeating pattern of string, int, string separated by a space

## Question 15

25 out of 40 points



CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles Simms

Assume that you have a CSV file which contains the information for each student and the scores for 4 homeworks. The structure for the data is - student ID, hw1, hw2, hw3, hw4. See the example below.

1, 20, 30, 28, 18

2, 35, 50, 27, 36

3, 17, 20, 34, 44

The struct definition is provided to you as follows:

```
typedef struct Student_struct
{
    int id;

    int hw1, hw2, hw3, hw4;
} Student;
```

Write the **code** that does the following:

(4 pts) Verify the number of command line arguments to make sure the user is running the program correctly. If not print out the error message "Usage: ./exeName inputFile.csv outputFile.csv" and "return -1".

(4 pts) Open the input file, provided in command line argument, for reading and check if it was opened successfully. If not, "return -1".

(4 pts) Go through the file once and calculate the number of lines in the file. Assume, there is NO header row.

(4 pts) Dynamically allocate the array for the Student struct.

(6 pts) Read the file again and store the data into the dynamically created

(2 pts) Read the file again and store the data into the dynamically created array above.

(2 pts) Close the input file being read.

(4 pts) Open the output file, provided in command line argument, for writing and check if it was opened successfully. If not, "return -1".

(6 pts) Go through the array and calculate the **AVERAGE** grade of the 4 homeworks, for each student. Write the student ID and the calculated value (up to 2 decimal places) into the output file. Following is the sample output file for the given input file example.

1,24.00

2,37.00

3,28.75

(2 pts) Close the output file.

(4 pts) Free the dynamically allocated array and "return 0" from the main function.

Assumptions:

- You are writing this code in the **main()** function. Write only the code to do the above. DO NOT create any other functions.
- All header files you would need are included.
- The struct definition is also written and included before the main function, so you can directly use it, you do not need write it.

CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles Simms

Selected /\*header file for reference

Answer: #ifndef STUDENT\_H

#define STUDENT\_H

typedef struct Student\_Struct{

int id;

int hw1, hw2, hw3, hw4;//4 homework assignment objs

}Student;

#endif\*/

//function for average--from main output as %.2lf

void average(int a, int b, int c, int d){

double avg = (a+b+c+d)/4;//20+30+28+18=96/4=24

}

//main function

#include <stdio.h>

#include <stdlib.h>

```

#include "student.h"
int main(int argc, char*argv[]){
    //vars
    int i = 0, count = 0;//for use in loop
    int avgGrade = 0;
    Student* array;//struct array to read csv data into
    array = (Student*)malloc(sizeof(Student*) * 10);
    //verify # of cmd args print error if incorrect
    if (argc < 3){
        printf("Usage: ./exeName inputFile.csv outputFile.csv");
        return 1;
    }
    //open input file check if NULL
    FILE*fileIn = fopen("inputFile.csv", "r");//read input
    if (fileIn == NULL){
        return 1;
        fclose(fileIn);
    }
    //read via loop get # of lines
    while (!feof(fileIn)){
        //read in by format
        fscanf(fileIn, "%[^,],%[^,],%[^,],%[^,],%[^,]\n",array.id[i],
array[i].hw1, array[i].hw2, array[i].hw3, array[i].hw4);
        count += count;//increment count of lines
        //rewind
        rewind(fileIn);//go back to beginning
        //fclose
        fclose(fileIn);
        //open output file
        FILE*fileIn = fopen("output.csv", "w");//write to output
        //traverse array to calc avg of 4 hws for each student
        if (fileIn == NULL){
            return -1;
        }
        for (i = 0; i < count; i++){

            average(array[i].hw1,array[i].hw2,array[i].hw3,array[i].hw4);//function
            call for average
            //write the student ID and value as double
            fprintf("%d, %.2lf\n", array[i].id,averageGrade[i]);
            //fclose
            fclose(fileIn);
            //free(array)
            free(array);
            return 0;
        }
    }
}

```

Correct [None]

Answer:

Response keeping brackets {} match in loop statements; should convert the  
Feedback: type of a, b, c, d to double or change 4 to 4.0 when calculating  
Avg; calculate the number of lines, allocate dynamic memory  
incorrectly; should calculate the number of lines and store data to  
the array in different loops; sscanf() need &; should use command  
line arguments as file names; incorrect fprintf();

## Question 16

22 out of 25 points



CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles  
Simms

Write the code for a `MAKEFILE`, considering that you have the following  
files: `main.c`, `mesh.h`, `mesh.c`, `point.h`, `point.c`

The dependencies/includes for each file are as below:

`main.c` -> `mesh.h`, `point.h`

`mesh.c` -> `mesh.h`, `point.h`

`point.c` -> `point.h`

Without the use of a makefile, you would compile the program as  
below:

```
gcc main.c mesh.c point.c -o myProg
```

CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles  
Simms

Selected myProg: `main.o mesh.o point.o`

Answer: `gcc -c main.o -o mesh.o -o point.o myProg`

`main.o: main.c mesh.h point.h`

`gcc -c main.c -o main.o`

`point.o: point.c point.h`

`gcc -c point.c -o point.o`

`clean:`

`rm -rf*.o`

`rm myProg`

Correct

Answer:

```
myProg: main.o mesh.o point.o
```

```
gcc main.o mesh.o point.o -o myProg
```

```
main.o: main.c mesh.h point.h
gcc -c main.c
mesh.o: mesh.c mesh.h point.h
gcc -c mesh.c
point.o: point.c point.h
gcc -c point.c
clean :
rm *.o
rm myProg
```

Response the make recipe for myProg should be: gcc main.o mesh.o  
Feedback: point.o -o myProg; need a space between -rf and \*.o

## Question 17

5 out of 25 points



CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles  
Simms

Write a **recursive function** that prints out all the **odd** numbers between **x** and **y** inclusive.

If **x** is even, then the print out will start at **x - 1**.

If **y** is even, then the last print out will be **y - 1**.

Assume **x** is always lower than **y**.

CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles  
Simms

Selected

Answer:

```
#include <stdio.h>
#include <stdlib.h>
```

```
//function to calculate odds
int odds(int n){
    int x, y;
    if (y > x){
        n = (y - x)/2;
        if (x % 2 == 0){
            x = x - 1;
        }
        else if (y % 2 == 0){
            y = y - 1;
        }
        n++;
        return n;
    }
}
//main function
int main(int argc, char*argv[]){
    int x = 0, y = 0;

    for (i = 0; i < y; i++){
        odds(i);
    }
    return 0;
}
// these codes confuse me without being able to verify
how it is working via testing it!
```

Correct  
Answer:

```
void printOdd( int x, int y )
{
    if( x % 2 == 0 )
        x--;

    if( x <= y )
        printf( "%d\n", x );

    if( x >= y )
        return;
    printOdd( x + 2, y );
}
```



Response  
Feedback:

nothing printed; no base case and recursive call





Write a **recursive function** to calculate and return the **PRODUCT** of the individual digits of an integer. Two examples below:

Input-1 => 687

Output-1 => 336

Input-2 => 12

Output-2 => 2

CS-1714-0B1-Fall-2020-Computer Programming II - aak759 - Rod Miles  
Simms

Selected

Answer:

```
int calculateProduct(int n){
    while (n > 0){
        if (n < 10){
            return n;//base case
        }
        else{
            return (n%10) * (n /10);//recursive case
        }
    }
}

int main(int argc, char*argv[]){
    .....
    //call from main function
    //int output = calculateProduct(<localVar>);
}

//these codes confuse me without being able to test
them!
```

Correct

Answer:

```
int prodDigits(int n)
{
    if(n == 0)
        return 1;
    else
        return n%10 * prodDigits(n/10);
}

```



Response

Feedback:

should return n%10\*calculateProduct(n/10) - no recursive call function name

Monday, December 7, 2020 9:43:44 PM CST

← OK

