Lab 5

- Due Oct 14 at 11:59pm
- Points 100
- Questions 15
- Available Oct 8 at 8am - Oct 14 at 11:59pm
- Time Limit None

Instructions

You are to answer the following questions. You are allowed to use your notes. You can work with other students. However, you will see some of this material on the final exam, therefore, I strongly suggest you do not simply copy answers from another student. This is your opportunity to make sure you understand this material.

This quiz was locked Oct 14 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	49 minutes	92 out of 100

Score for this quiz: 92 out of 100
Submitted Oct 8 at 2:25pm
This attempt took 49 minutes.

Question 1
8 / 8 pts

This is a 3 part questions. When answering the questions you MUST indicate which part you are completing. (Label your answer with Part 1, Part 2, and Part 3.)

Part 1: (4 points)

Create a struct named ‘Data’ that has 1 short, 1 float, 1 character, and 2 integers. When creating the struct, order the data using the rule to minimizes the amount of padding that will be needed.

Part 2: (3 points)

Now assume you need to allocated memory for “N” number of the struct you defined in the box above. Write the code to dynamically allocated the memory using the “C” function **calloc**.

Part 3: (1 point)

Now write the code to give the memory back to the operating system.

Your Answer:

Part 1:

```
struct Data {

int integer1;

int integer2;

float decimal;

short shortValue;

char character;

};
```

Part 2:

```
struct Data* dataArray = (struct Data*) calloc(N, sizeof(struct Data));
```

Part 3:

free(dataArray);

Question 2
10 / 10 pts

This is a 2 part question: When answering the questions you MUST indicate which part you are completing. (Label your answer with Part 1, and Part 2.)

Part 1: 7 points

Assume you have a struct called **Pixel.**

The struct will represent one pixel of an image. The size of the image is 200 X 400 Pixels.

#define ROWS 200;

#define COLS 400;

In class we covered at least 3 different algorithms on how to dynamically allocate memory for a 2D array.

You are going to write the code to dynamically allocate memory for the pixels for a 2D image of size ROWS and COLS. Use the algorithm illustrated in the skit we did in class. HINT: allocating the memory for the students volunteers in the front of the class (the pointers), then allocating the memory for each set of students in the audience (the data). This will require multiple calls to malloc. All memory must be stored on the **Heap not the Stack**. **REMEMBER THE DATA TYPE IS A PIXEL.**

Part 2: 3 points

Write the code that gives the memory back to the operating system.

REMEMBER TO LABEL YOUR ANSWERS: PART1 AND PART 2

Your Answer:

Part 1:

```
struct Pixel {
\\ some members of the pixel struct
}
#define ROWS 200
#define COLS 400
struct Pixel** image = (struct Pixel**)malloc(ROWS * sizeof(struct Pixel*));
for (int I = 0; i < ROWS; ++i) {
image[i] = (struct Pixel*)malloc(COLS * sizeof(struct Pixel));
}
```

Part 2:

```
for (int I = 0; I < ROWS; ++i) {
free(image[I]);
}

free(image);
```

Question 3
3 / 3 pts

Assume you have already dynamically allocated memory for a 1D array of integers, using row, col, and sizeof(int). Now suppose, you want to access the data as if you were accessing a 2D array. In class we discussed an algorithm that allows us to access the 1D array using a formula consisting of variables, such as, **current_row, current_col,** and **total_width.** This formula will allow us to think of the 1D array in terms of a 2D array.

Using the variables listed (**current_row, current_col and total_width**), fill in the formula that would allow access to elements of the 1D array as if it were a 2D array.

arr[**What would go here**] = numVal;

You Answered

(current_row * total_width) + current_col

Correct Answers

(current_row*total_width)+current_col

Question 4

3 / 3 pts

In Chapter 1, we learned that information in a system is represented as a bunch of bits. As an example: A particular sequence of bytes (bits) could represent a range of different type of data. The thing that distinguishes the different data is the _____ in which we view/use the data. This question is directly out of the notes.

Look in the chapter 1 at slide 5 .

Correct!

context

Correct Answers

context

Question 5

4 / 8 pts

This is a 2 part question. **Label your answers with Part 1 and Part 2.**

Part 1 (4 points)

Given the following struct:

```
struct Test{
        char a;
        int b;
        short c;
        char d;
        int e;
};
```

What is the size of this struct, in bytes?

Part 2: (4 points)

Rewrite the struct using the rule, discussed in class, that will minimize the amount of padding needed, also pushing the padding to the end of the memory.

Your Answer:

Part 1:

struct Test {

char a;

int b;

short c;

char d;

int e;

}; (THIS IS 20 BYTES)

Part 2:

struct Test {

int b;

int e;

short c;

char a;

char d;

}; (THIS IS 16 BYTES)

part 1 answer is 16bytes

Question 6
4 / 4 pts

During the compile process, the #define MAX_SIZE, in the program below, is stored on the stack.

```
#include <stdio.h>

#define MAX_SIZE 10

 int main(){

    int size = 3;

    size = size + MAX_SIZE;

    printf(“The value of size is %d\n”, size);

    return 0;

}
```

- True

Correct!

- False

Question 7
8 / 8 pts

What is the output of the following program?

```
#include <stdio.h>

int f(int , int *, int **);

int  main(){

   int c, *b, **a;

   c = 12;

   b = &c;
```

```
    a = &b;


    printf("%d \n", f(c, b, a));
    return 0;
}
int f(int x, int *py, int **ppz){
  int y, z, i;
  **ppz += 3;
   z  = **ppz;
  for(i = 0; i < (x-6); i++
       *py += 4;
  y=*py;
  x += 9;
  return x + y + z;
}
```

Correct!

75

Correct Answers

75

Question 8

6 / 6 pts

What is the output of the following program?

```
#include <stdio.h>

int main()
{
    float arr[10] = {70.5, 0.5, 17.8, 5.9, 19.5, 6.3, 8.2, 5.6, 2.1, 6.4};
    float *ptr1 = &arr[1];
    float *ptr2 = ptr1 + 6;

    printf("%.2f\n", *ptr2);
    printf("%ld\n", ptr2 - ptr1);

   return 0;
}
```

Your Answer:

5.60

6

Question 9
4 / 4 pts

One possible advantage of using size_t is, it is always positive.

Correct!

- (●) True
- ( ) False

Question 10
8 / 8 pts

Assume we are compiling a **hello.c** file:

In order from **start to finish**, name each phase of the compilation system and the file each phase will produce.

PHASE	FILE PRODUCED
preprocessing	hello.i (preprocesse
compilation	hello.s (assembly c
assembly	hello.o (object)
linking	hello (executable)

Answer 1:
Correct! Preprocessing
Correct Answer
pre-processing
Correct Answer
Pre-Processing
Correct!

Preprocessing

Correct Answer

Pre-processing

Answer 2:

You Answered hello.i (preprocessed file)

Correct Answer

hello.i

Correct Answer

Hello.i

Correct Answer

.i

Answer 3:

You Answered compilation

Correct Answer

Compiler

Correct Answer

COMPILER

Correct Answer

COMPILE

Correct Answer

Compile

Correct Answer

compiler

Correct Answer

compile

Answer 4:

You Answered hello.s (assembly code)

Correct Answer

hello.s

Correct Answer

Hello.s

Correct Answer

.s

Answer 5:

You Answered assembly

Correct Answer

Assembler

Correct Answer

ASEMBLER

Correct Answer

assembler

Correct Answer

Assemble

Correct Answer

assemble

Answer 6:

You Answered hello.o (object)

Correct Answer

hello.o

Correct Answer

HELLO.o

Correct Answer

Hello.o

Correct Answer

.o

Answer 7:

You Answered linking

Correct Answer

Linker

Correct Answer

LINKER

Correct Answer

linker

Correct Answer

link

Correct Answer

Link

Answer 8:

You Answered hello (executable)

Correct Answer

.o

Correct Answer

executable

Correct Answer

a.out

Correct Answer

hello

Correct Answer

hello.o

Correct Answer

Hello.o

Correct Answer

Executable

Question 11

8 / 8 pts

The following program does not have the output that I was expecting:

Output:

Value of c before change: 0x7ff7be93c5e8

Value of d: 0x7ff7be93c5e4

Value of c after change: 0x7ff7be93c5e8 This is Not What I was expecting

Expected output:

Value of c before change: 0x7ff7be93c5e8

Value of d: 0x7ff7be93c5e4

Value of c after change: 0x7ff7be93c5e4 This is what I was expecting

Determine what will fix this program so that after the call to change, ‘c’ will point to the same thing ‘d’ points to. You are not allowed to add lines of code, simply list the line number and the necessary change. Don’t over think this. There are 4 minor changes needed to fix this code.

REWRITE THE 4 LINES OF CODE NEEDED TO FIX THIS PROBLEM. INDICATE WHICH LINE YOU ARE CHANGING BY THE NUMBER I PLACED BY EACH LINE OF CODE.

```
1. #include <stdio.h>
2. void change(int *, int *);
3. int main(){
4.     int a = 10;
5.     int b = 20;
6.     int *c = &a;
7.     printf("Value of c before change: %p \n", c);
8.     int *d = &b;
```

```
9.    printf("Value of d: %p \n", d);
10.   change(c, d);
11.   printf("Value of c after change: %p \n", c);
12.   return 0;
13. }
14.  void change(int * x, int * z){
15.      x = z;
16.  }
```

Your Answer:

Line 2: void change(int **, int *);

Line 10: change(&c, d);

Line 14: void change(int **x, int *z)

Line 15: *x = z;

Question 12
6 / 6 pts

In class we discussed three things that make up the signature of a function. This is important when creating a function pointer. List the three things that defines a function signature.

Your Answer:

Function name

Parameter list/type

Return type

Question 13
0 / 4 pts

Cache memory is smaller, faster, and cheaper memory (financially speaking). It serves as temporary staging areas for information that the processor is likely to need in the near future.

You Answered

- True

Correct Answer

- False

Question 14
10 / 10 pts

In order, **highest address to lowest address**: list the 5 areas of the Virtual (memory) address space for the Linux OS that we discussed in class.

Your Answer:

Kernel Virtual Memory

User Stack

Shared Libraries

Heap

Program Code and Data

Question 15
10 / 10 pts

This is a 4 part question. LABLE YOUR ANSWER WITH THE APPROPRIATE NUMBER.

If given the following function prototypes:

float plus(float, float);

float minus(float, float);

float multiply(float, float);

float divide(float, float);

PART 1:

Create a function pointer that can be pointed to one of the 4 functions above.

PART 2:

Now point the function pointer (created above) to one of the four functions above.

PART 3:

Write the code to call the function pointer created and initialized in “a” and “b” above (you can make up data values).

PART 4:

Create a second function pointer and initialize it in one line of code.

Your Answer:

Part 1:

float (*funct_ptr)(float, float);

Part 2:

```
func_ptr = &plus;
```

Part 3:

```
float result = func_ptr(5.0, 3.0);
printf("Result: %.2f/n", result);
```

Part 4:

```
float (*func_ptr2)(float, float) = multiply;
```

Quiz Score: 92 out of 100