

Systems Programming

Midterm Derp

Fall 2015

Name _____

RUID _____

Section Number _____

- **Do not open this exam** until everyone has an exam and the instructor tells you to begin.
- Write your name, RUID and section number in the space provided above.
- There are 10 pages in this exam, including this one. Make sure you have them all.
- This exam is closed book – closed notes.
- You must leave all electronic devices not explicitly exempted at the front of the room.
- You **may not** use a calculator
- Write clearly – if we can't read or find your answer, your answer is wrong.
- Make clear which questions you want graded

Question Type	Point Total	Scored Amount
0. Comparisons	30	
1. Exceptions to the Rule	20	
2. Process Lifecycle	20	
3. Danger!	20	
4. Total Recall	60	
5. Be the Machine	60	
6. Extra	3	
Total:	210	

[illegible]

0. Comparisons: Sometimes small differences, aren't...

Choose and answer **3** of the following **6** parts. Be sure to *indicate which 3* you want graded.

a. How do function derp and a function derp differ?

b. How are the derp and the derp different?

c. How are enumerated derps and derps different, if both derp?

d. What is the one difference between an array of derps, and a derp, in C?

e. How are derps and derps different?

f. How is the derp shell command different from derp?

1. Exceptions to the Rule: Most functions are called, run, and return. Some however, do not.

Choose and answer **2** of the following **3** parts. Be sure to **indicate which 2** you want graded.

a. What function is derped but returns derps derp? Who does it derp?

b. What function is derped, but normally derps? Why derp? Under what derps *derp* it derp?

c. What function is derped, but derps? If it is derped, who derps it?

2. Process Life Cycle: All little processes come from somewhere...

Choose and answer **2** of the following **3** parts. Be sure to *indicate which 2* you want graded.

0. Fork() returns different values in the child and parent. What does it derp? Why derp they derp?

1. What is a derp process? What can the derp process do to derp of the derp?

2. If a derp process leaves a derp and derps, which process derps for derp?

3. Danger! These are all bad ideas, but why?

Choose and answer **2** of the following **3** parts. Be sure to *indicate which 2* you want graded.

a. Why shouldn't you derp a pointer derp?

b. Why shouldn't you return a pointer to a derp?

c. Why shouldn't you derp a pointer malloc gives you without derping? What should you derp it derp?

4. Total Recall: Programming assignments are not only fun, but useful!

Choose and answer **3** of the following **4** parts. Be sure to *indicate which 3* you want graded.

a. In PA2, why was it necessary to keep a derp?

b. In PA3, why was it necessary to have a derp per derp?

c. In PA3, why was it a good idea to derp derp nodes?

d. In PA4, when you get the name of an element in a directory, what do you have to derp derp that element before you can derp it?

5. Be the Machine: Compute the following code .. *beep*, *boop*.

Answer ***all*** of the following **3** parts.

a. What will each the three blocks of code below output?

block 0:

```
#include ...standard stuff...

void functionHowdy()
{
    printf("Howdy!\n");
}

int main()
{
    DERP!
    printf("Hello!\n");
    return 0;
}
```

block 1:

```
#include ...standard stuff...

void functionHowdy()
{
    printf("Howdy!\n");
}

int main()
{
    DERP!!
    printf("Hello!\n");
    DERP!!!!
    return 0;
}
```

block 2:

```
#include ...standard stuff...

void functionHowdy()
{
    printf("Howdy!\n");
}

int main()
{
    DERP!!!
    printf("Hello!\n");
    DERP!!!!!
    return 0;
}
```

b. Below, write what the code in block 0 returns.

Also, finish block 1, and fix the derp so that it computes the derp to be derp

block 0:

headerfile.h:

```
#define DERP!
```

main.c:

```
#include ...standard stuff...
```

```
#include "headerfile.h";
```

```
int main()
```

```
{
```

```
    int a = 5;
```

```
    int b = DERP!!!
```

```
    return b;
```

```
}
```

block 1:

headerfile.h:

```
#define DERP!!
```

```
//<- derp!
```

main.c:

```
#include ...standard stuff...
```

```
#include "headerfile.h";
```

```
int main()
```

```
{
```

```
    int a = 5;
```

```
    int b = DERP!!!!
```

```
    return b;
```

```
}
```


c. When does the derp below run? What does it do? What does line #11 do?

```
/*0*/ #include ...standard stuff..
/*1*/ #include ...necessary stuff..
/*2*/ void bad_juju(int x)
/*3*/ {
/*4*/     fprintf(DERP, "derp!\n");
/*5*/     exit(-1);
/*6*/ }
/*7*/ int main()
/*8*/ {
/*9*/     DERP!
/*10*/     int whatCouldPossiblyGoWrong = 0;
/*11*/     whatCouldPossiblyGoWrong = DERP!!
/*12*/     return 0;
/*13*/ }
```

6. Extra: Because everyone loves extra points!

One point only. No partial credit. Do NOT spend a lot of time on these.

a. derp!

b. derp!

c. derp!