ECE264 Fall 2017
Exam 1, 8-930PM, September 12

In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action.

Signature:

You must sign here. Otherwise you will receive a **1-point** *penalty.*

Read the questions carefully.
Some questions have conditions and restrictions.

This is an *open-book, open-note* exam. You may use any book, notes, or program printouts. No electronic device is allowed. You may **not** borrow books from other students.

This exam tests one learning objective:
Structure (Question 2 and 3)
You must obtain 50% or more points in each of the corresponding question to pass the learning objective.

Remove the top sheet.
Write your answers on page 2.
Return only the top sheet.

Contents

Total Score:

Learning Objective

Structure Pass Fail

Q1	
A	
B	
C	
D	
E	

Q2	
A	
B	
C	
D	
E	

Q3	
A	
B	
C	
D	
E	
F	

Q4	
A	`s =  t =`
B	`s =  t =`

Q5	
A	
B	

This page is blank.

1 GDB (10 points)

Each question is 2 points.
Consider the following program:

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 int f1(int a)
 4 {
 5   int b = a + 1;
 6   return b;
 7 }
 8
 9 int f2(int a)
10 {
11   if (a == 0)
12     {
13       return f1(a + 2);
14     }
15   int b = a;
16   int c = 0;
17   c = b + f2(a - 1);
18   return c;
19 }
20
21 int main(int argc, char * * argv)
22 {
23   int a = 0;
24   a = f2(3);
25   printf("a = %d\n", a);
26   return EXIT_SUCCESS;
27 }
```

Suppose you run the following GDB commands; `(gdb)` is the prompt.

```
(gdb) b main
Breakpoint 1 at 0x400593: file q1.c, line 23.
(gdb) b f1
Breakpoint 2 at 0x40052d: file q1.c, line 5.
(gdb) r
Breakpoint 1, main (argc=1, argv=0x7fffffffe328) at q1.c:23
23   int a = 0;
(gdb) c
```

The output from `gdb` is

```
Breakpoint 2, A (a=B) at q1.c:5

(gdb) bt
```

The output from `gdb` is

```
# 0 f1(a=2) at q1.c:5
# 1 0x0000000000400559 in f2 (a=0) at q1.c:13
# 2 Cin f2 (a=D) at q1.c:E
# 3 0x0000000000400575 in f2 (a=2) at q1.c:17
# 4 0x0000000000400575 in f2 (a=3) at q1.c:17
# 5 0x00000000004005a4 in main (argc=1, argv=0x7fffffffe328) at q1.c:24
```

Please write your answers for A, B, C, D, and E on **page 2**.
Answer:

```
A: f1
B: 2
C: 0x0000000000400575
D: 1
E: 17
```

2 Structure (20 points)

Each question is 4 points.
Please write the code for `FIX ME`.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <math.h>
 4 #pragma pack(1) // tell compiler not to pad any space
 5 // assume sizeof(char) = 1, sizeof(int) = 4,
 6 // sizeof(double) = 8, sizeof(a pointer) = 8
 7 typedef double (* funcptrtype) (int, int);
 8 typedef struct
 9 {
10   int x;
11   int y;
12   double length;
13   funcptrtype fptr;
14 } Vector;
15
16 double length1(int x, int y)
17 {
18   return (double)(x * x + y * y);
19 }
20
21 double length2(int x, int y)
22 {
23   double lensq = length1(x, y);
24   return sqrt(lensq);
25 }
26
27 void Vector_Length(Vector * vptr)
28 {
29   funcptrtype fp;
30   // <--- FIX ME ---> A
31   // assign vptr's fptr to fp
32
33
34   // <--- FIX ME ---> B
35   // use fp to call a function with two arguments:
36   //     vptr's x as the first argument
37   //     vptr's y as the second argument
38   // the result is stored in vptr's length
```

```
39
40
41 }
42
43 void Vector_print(Vector * vptr)
44 {
45   printf("x = %d\n", vptr -> x);
46   printf("y = %d\n", vptr -> y);
47   printf("l = %f\n", vptr -> length);
48 }
49
50 int main(int argc, char ** argv)
51 {
52   Vector v1;
53   v1.x = 3;
54   v1.y = 6;
55   // <--- FIX ME ---> C
56   // create a pointer of Vector
57   // This pointer is called vptr
58
59
60   // <--- FIX ME ---> D
61   // assign v1's address to vptr's value
62
63
64   // <--- FIX ME ---> E
65   // create an array
66   // The array's type is funcptrtype
67   // The array's name is fparr
68   // The array has two elements called length1 and length2
69
70
71   for (int iter = 0; iter < 2; iter ++)
72     {
73       v1.fptr = fparr[iter];
74       Vector_Length(vptr);
75       Vector_print(vptr);
76       printf("----------------\n");
77     }
78
79   return EXIT_SUCCESS;
80 }
```

Answer:

```
fp = vptr -> fptr;
vptr -> length = fp(vptr -> x, vptr -> y);
Vector * vptr;
vptr = & v1;
funcptrtype fparr[] = {length1, length2};
```

3 Stack Memory and Structure (30 points)

Each answer is 5 points.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 typedef struct
 4 {
 5   int x;
 6   int y;
 7 } Vector;
 8
 9 void Vector_print(Vector v)
10 {
11   printf("v.x = %d, v.y = %d\n", v.x, v.y);
12 }
13
14 Vector Vector_double (Vector * vptr)
15 {
16   Vector v2;
17   v2 = * vptr;
18   (vptr -> x) *= 2;
19   (vptr -> y) *= 2;
20   return v2;
21 }
22
23
24 int main(int argc, char ** argv)
25 {
26   Vector v1;
27   v1.x = 3;
28   v1.y = 6;
29   Vector v2;
30   v2 = Vector_double(& v1);
31   Vector_print(v1);
32   Vector_print(v2);
33   return EXIT_SUCCESS;
34 }
```

Please fill the stack when the program has finished line 19 and before starting line 20.
Each address has `A` as the prefix.
DC means "do not care". You do not need to answer.

Frame	Symbol	Address	Value
Vector_double	v2.y	A212	*DC*
	v2.x	A208	Question A
	vptr	A200	Question B
	value address		Question C
	return location: line		Question D
main	v2.y	A124	*DC*
	v2.x	A120	*DC*
	v1.y	A116	Question E
	v1.x	A112	Question F
	argv	A104	*DC*
	argc	A100	*DC*

Answer:

```
A: 3
B: A112
C: A120
D: line 31
E: 12
F: 6
```

4 Pointer (12 points)

Write the outputs of this program.
Each answer is 3 points.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 typedef void (* swapptr)(int *, int *);
 4
 5 void swap1(int * a, int  * b)
 6 {
 7   int * s;
 8   s = a;
 9   b = s;
10   a = b;
11 }
12
13 void swap2(int * a, int  * b)
14 {
15   int * s;
16   s = a;
17   * a = * b;
18   b = s;
19 }
20
21 int main(int argc, char * * argv)
22 {
23   swapptr swaparray [] = {swap1, swap2};
24   int numswap = sizeof(swaparray) / sizeof(swapptr);
25   for (int ind = 0; ind < numswap; ind ++)
26     {
27       swapptr func = swaparray[ind];
28       int s = 2;
29       int t = 6;
30       func(& s, & t);
31       printf("s = %d, t = %d\n",  s,  t);
32       // result from swap1 is question A
33       // result from swap2 is question B
34     }
35   return EXIT_SUCCESS;
36 }
```

Answer:

```
s = 2, t = 6
```

```
s = 6, t = 6
```

5 Number Conversion (8 points)

The following program has one (or several) mistake. What is the output of this program? **DO NOT** correct the program. The purpose of this question is to show that some test cases may reveal coding mistakes while some other test cases cannot.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 void print_digit(int n, int radix)
 4 {
 5   if (n < 10)   {   fputc(n + '0', stdout);   }
 6   else    {   fputc(n - 10 +  'a', stdout);   }
 7 }
 8 void print_integer(int n, int radix)
 9 {
10   if (n < radix)
11     {
12       print_digit(n, radix);
13       fputc('\n', stdout);
14       return;
15     }
16   // find the largest power of the radix smaller than n
17   int p = radix;
18   while  ((p * radix) <= n)
19     {
20       p = p * radix;
21     }
22   while (n > 0) // <--- WRONG ---> should be (p > 0)
23     {
24       print_digit(n / p, radix);
25       n -= (n / p) * p;
26       p = p / radix;
27     }
28   fputc('\n', stdout);
29 }
30
31 int main(int argc, char * * argv)
32 {
33   print_integer(11, 14); // answer A
34   print_integer(225, 15); // answer B
35   return EXIT_SUCCESS;
36 }
```

Answer:

```
b
1
```