ECE264 Spring 2016
Exam 1, 630-730PM, February 11

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 personal electronic device is allowed. You may **not** borrow books from other students.

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

Contents

Learning Objective

Recursion	Pass	Fail
Structure	Pass	Fail

The ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
00	00	NUL	32	20	SP	64	40	@	96	60	‘
01	01	SOH	33	21	!	65	41	A	97	61	a
02	02	STX	34	22	"	66	42	B	98	62	b
03	03	ETX	35	23	#	67	43	C	99	63	c
04	04	EOT	36	24	$	68	44	D	100	64	d
05	05	ENQ	37	25	%	69	45	E	101	65	e
06	06	ACK	38	26	&	70	46	F	102	66	f
07	07	BEL	39	27	’	71	47	G	103	67	g
08	08	BS	40	28	(	72	48	H	104	68	h
09	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

1 Call Stack (4 points)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  char f1(int x, int y, int * z)
4  {
5    char t = x + y + '0';
6    * z = x - y;
7    return t;
8  }
9
10 int main(int argc, char * * argv)
11 {
12   char t = -9;
13   int x = 5;
14   if (argc < 2)
15     {
16       return EXIT_FAILURE;
17     }
18   t = f1(argv[1][0] - argv[1][1], argc, & x);
19   // L1
20   printf("%c %d\n", t, x);
21   return EXIT_SUCCESS;
22 }
```

The program's name is q1.
The program is called in a terminal using
`./q1 ece264 Purdue 2016`

Please fill this table as the program progresses.
'-' means do not worry about it.
Please clearly distinguish '0' from 0 in your answers.

A.		F.	
B.		G.	
C.		H.	
D.		I.	
E.		J.	

When `f1` is called, just before executing line 5.

Frame	Symbol	Address	Value
f1	t	MN070	-
	z	MN060	A.
	y	MN050	-
	x	MN040	B.
	Value Address		C.
	Return Location		D.
main	x	MN030	5
	t	MN020	-9
	argv	MN010	-
	argc	MN000	E.

After line 5, before line 6:

Frame	Symbol	Address	Value
f1	t	MN070	F.
	z	MN060	-
	y	MN050	-
	x	MN040	-
	Value Address		-
	Return Location		-
main	x	MN030	5
	t	MN020	-9
	argv	MN010	-
	argc	MN000	-

After line 6, before line 7

Frame	Symbol	Address	Value
f1	t	MN070	-
	z	MN060	G.
	y	MN050	-
	x	MN040	-
	Value Address		-
	Return Location		-
main	x	MN030	H.
	t	MN020	-
	argv	MN010	-
	argc	MN000	-

Function `f1` finishes

Frame	Symbol	Address	Value
main	x	MN030	I.
	t	MN020	J.
	argv	MN010	-
	argc	MN000	-

2 Recursion (4 points)

What is the output of this program?

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

3 Structure (4.5 points)

Please fill the code.
For your reference, the following shows a few instances of execution.

```
$ gcc -Wall -Wshadow q3prog.c -c q3prog

$ ./q3prog 0 1.1 0.99
numfunc = 3
funcindex = 0
str1.z = 1

$ ./q3prog 0 1.1 2.2
numfunc = 3
funcindex = 0
str1.z = 0

$ ./q3prog 2 1.1 2.2
numfunc = 3
funcindex = 2
str1.z = 0

$ ./q3prog 2 1.5 2.2
numfunc = 3
funcindex = 2
str1.z = 1

$ ./q3prog 1 1.5 1.2
numfunc = 3
funcindex = 1
str1.z = 1

$ ./q3prog 1 1.5 1.3
numfunc = 3
funcindex = 1
str1.z = 0
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // create a type for function pointers
5 // the function takes two double as arguments
6 // and returns one integer
```

```
 7 // the type is called "functype"
 8 // --->>> FILL THE CODE HERE <<<---
 9
10 typedef
11
12
13 // create a structure that has the following attributes
14 // a function pointer
15 //       the arguments are two doubles
16 //       return type is an integer
17 // two double
18 // one integer
19
20 // --->>> FILL THE CODE HERE <<<---
21
22 typedef
23
24 {
25   // a function pointer as an attibute
26   // --->>> FILL THE CODE HERE <<<---
27
28
29
30
31   double x;
32   double y;
33   int z;
34 } Q3Structure;
35
36 // call the function pointed by str's function pointer
37 // the arguments are str's two double attributes
38 // return the value calling that function using the arguments
39 int Q3StructZ(Q3Structure str)
40 {
41   // --->>> FILL THE CODE HERE <<<---
42
43
44
45
46   return
47 }
48
```

```
49 // create three functions
50 //       the arguments are two doubles
51 //       return type is an integer
52
53 int f0(double a, double b)
54 {
55   if (a > b)
56     { return 1; }
57   return 0;
58 }
59
60 int f1(double a, double b)
61 {
62   if (a > (b * b))
63     { return 1; }
64   return 0;
65 }
66
67 int f2(double a, double b)
68 {
69   if ((a * a) > b)
70     { return 1; }
71   return 0;
72 }
73
74 int main(int argc, char * * argv)
75 {
76   // main needs three arguments:
77   // argv[1] must be 0, 1, or 2 to select the functions
78   // argv[2] and argv[3] must be double as the arguments
79   if (argc < 4)
80     {
81       return EXIT_SUCCESS;
82     }
83
84   // create an array of functions
85   // the array is called funcarray
86   // the array has three elements: f0, f1, f2
87   // --->>> FILL THE CODE HERE <<<---
88
89
90
```

```
 91
 92
 93    // calculate the size of the array
 94    int numfunc = sizeof(funcarray) / sizeof(functype);
 95    printf("numfunc = %d\n", numfunc);
 96
 97    int funcindex = atoi(argv[1]);
 98    if ((funcindex < 0) || (funcindex >= numfunc))
 99      {
100        // invalid index
101        return EXIT_FAILURE;
102      }
103    printf("funcindex = %d\n", funcindex);
104    // create a function pointer called funcptr
105    functype funcptr;
106    // assign the function
107    funcptr = funcarray[funcindex];
108
109    // create a structure variable
110    Q3Structure str1;
111    str1.ptr = funcptr;
112    str1.x = atof(argv[2]);
113    str1.y = atof(argv[3]);
114    // assign the z value
115    str1.z = Q3StructZ(str1);
116    printf("str1.z = %d\n", str1.z);
117    return EXIT_SUCCESS;
118 }
```