UNIVERSITY OF CALIFORNIA, LOS ANGELES UCLA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

CS 33

Midterm #2

All answers must be written on the answer sheet (last page of the exam).

All work should be written directly on the exam, use the backs of pages if needed.

This is an open book, open notes quiz – but you cannot share books or notes. An ASCII table is on the second to last page if you need it.

I will follow the guidelines of the university in reporting academic misconduct – please do not cheat.

NAME: SOLUTION _____

ID: _____

Problem 1: _____

Problem 2: _____

Problem 3: _____

Total: _____

1. ***Structured Play (30 points):*** Consider the following declaration:

```
struct node_t {
  char color[6];
  union {
    int numeric;
    char label[8];
  } identifier;
  long value;
} a[4];
```

color is 6 bytes in size no alignment, but we must pad with 2 bytes to meet alignment rules for the next member.

identifer is 8 bytes in size, with int aligment (must begin at multiple of 4).

value is 8 bytes in size, with type long alignment (for x86_64 must begin at mulitple of 8).

Overall alignment of node_t must be on multiples of 8.

Answer the following questions on how this data structure would be laid out on a 64-bit Linux machine:

a) Considering alignment – how much total space (in bytes) would this data structure require?

Total size = 4*sizeof(*node_t*) = 4*24 bytes = 96 bytes

b) If the base address of array *a* is 0x600a60, what would be the string stored in *a[1].color*? Make use of the following gdb output:

```
(gdb) x/24x 0x600a60
0x600a60:       0x7675616d      0x00000a65      0x00000400      0x00000000
0x600a70:       0x46e87ccd      0x00000000      0x68636f6d      0x00000a61
0x600a80:       0x00000800      0x00000000      0x3d1b58ba      0x00000000
0x600a90:       0x72757a61      0x00000a65      0x00001000      0x00000000
0x600aa0:       0x507ed7ab      0x00000000      0x7268636f      0x00000a65
0x600ab0:       0x00000200      0x00000000      0x2eb141f2      0x00000000
```

6d 6f 63 68 61 0a = mocha

Hint – don't forget that gdb reverses byte ordering within each 4-byte chunk. So in the following dump:

```
(gdb) x/4x 0x00111110
0x111110:   0x33221100  0x77665544  0xBBAA9988  0xFFEEDDCC
```

This prints out 16 bytes of memory starting at address 0x111110. In this example, the 16 bytes of memory starting at 0x111110 would contain, in order from lowest address (0x111110) to highest address (0x11111F):

00112233445566778899AABBCCDDEEFF

So address 0x111110 contains the byte 0x00, address 0x111111 contains the byte 0x11, address 0x111112 contains the byte 0x22, and so on. So in terms of just the least significant hex place of the address, gdb is actually printing out addresses in the following order:

3 2 1 0 7 6 5 4 B A 9 8 F E D C

This is useful when reading words, but can be confusing for other values.

b) If the base address of array *a* is 0x600a60, what would be the string stored in *a[1].color*? Make use of the following gdb output:

```
(gdb) x/24x 0x600a60
0x600a60:       0x68636f6d      0x00000a61      0x00000400      0x00000000
0x600a70:       0x46e87ccd      0x00000000      0x7268636f      0x00000a65
0x600a80:       0x00000800      0x00000000      0x3d1b58ba      0x00000000
0x600a90:       0x7675616d      0x00000a65      0x00001000      0x00000000
0x600aa0:       0x507ed7ab      0x00000000      0x72757a61      0x00000a65
0x600ab0:       0x00000200      0x00000000      0x2eb141f2      0x00000000
```

6f 63 68 72 65 0a = ochre

b) If the base address of array *a* is 0x600a60, what would be the string stored in *a[1].color*? Make use of the following gdb output:

```
(gdb) x/24x 0x600a60
0x600a60:       0x68636f6d      0x00000a61      0x00000400      0x00000000
0x600a70:       0x46e87ccd      0x00000000      0x72757a61      0x00000a65
0x600a80:       0x00000800      0x00000000      0x3d1b58ba      0x00000000
0x600a90:       0x7675616d      0x00000a65      0x00001000      0x00000000
0x600aa0:       0x507ed7ab      0x00000000      0x7268636f      0x00000a65
0x600ab0:       0x00000200      0x00000000      0x2eb141f2      0x00000000
```

61 7a 75 72 65 0a = azure

b) If the base address of array *a* is 0x600a60, what would be the string stored in *a[1].color*? Make use of the following gdb output:

```
(gdb) x/24x 0x600a60
0x600a60:       0x68636f6d      0x00000a61      0x00000400      0x00000000
0x600a70:       0x46e87ccd      0x00000000      0x7675616d      0x00000a65
0x600a80:       0x00000800      0x00000000      0x3d1b58ba      0x00000000
0x600a90:       0x72757a61      0x00000a65      0x00001000      0x00000000
0x600aa0:       0x507ed7ab      0x00000000      0x7268636f      0x00000a65
0x600ab0:       0x00000200      0x00000000      0x2eb141f2      0x00000000
```

6d 61 75 76 65 0a = mauve

2. ***Complete Dis-Array (30 points)***: Consider the following C fragment:

#define SIZE 10

int red[SIZE][SIZE];	*int orange[SIZE][SIZE];*	*int gold[SIZE][SIZE];*	*int black[SIZE][SIZE];*
*int *blue;*	*int *green;*	*int *bronze;*	*int *grey;*
*int *green[SIZE];*	*int *purple[SIZE];*	*int *silver[SIZE];*	*int *white[SIZE];*
int n;	*int n;*	*int n;*	*int n;*

int main(int argc, const char argv[]) {*
…

The code in …'s will create three two dimensional arrays: red, green, and blue. Arrays red and green are statically sized (i.e. size is known at compile time). Array red is a nested array and array green is a multi-level array. Array blue is dynamically sized (i.e. a dynamically nested array).

Each array has a function that sets the value of one element of the array. These functions have three parameters – the row position in the array (i), the column position in the array (j), and the value to set (val). They have the following prototypes:

void setred(int i, int j, int val);	*void setorange*	*void setgold*	*void setblack*
void setblue(int i, int j, int val);	*void setgreen*	*void setbronze*	*void setgrey*
void setgreen(int i, int j, int val);	*void setpurple*	*void setsilver*	*void setwhite*

For example, *setred(i,j,val)* will set *red[i][j]* to the value *val*.

Based on the information above, answer the following questions:

a) Which of these three functions (setred, setblue, or setgreen), (setorange, setgreen, or setpurple), (setgold, setbronze, or setsilver), (setblack, setgrey, or setwhite) is shown disassembled below?

```
8048450:       55                      push   %ebp
8048451:       8b 15 d4 9a 04 08       mov    0x8049ad4,%edx    blue
8048457:       89 e5                   mov    %esp,%ebp
8048459:       8b 45 08                mov    0x8(%ebp),%eax    i
804845c:       0f af 05 d0 9a 04 08    imul   0x8049ad0,%eax    i*n
8048463:       8b 4d 10                mov    0x10(%ebp),%ecx   val
8048466:       03 45 0c                add    0xc(%ebp),%eax    j
8048469:       89 0c 82                mov    %ecx,(%edx,%eax,4) blue[i*n+j]=val
804846c:       5d                      pop    %ebp
804846d:       c3                      ret
```

b) In the disassembled code above, what variable is located at address 0x8049ad0?

n

3. ***We Want the Func (40 points):*** Consider the following C code:

```
int foo(int i, int count) {

  if (count==32)
    return 0;
  else
    return (foo(i>>1,count+1)+(i&1));
}

int main( int argc, const char* argv[] ) {
  fprintf(stderr, "%d\n", foo(atoi(argv[1]), 0));
}
```

a) If this code were compiled, and then executed with the value 10, 11, 15, 31 as a parameter, what would it return?

par	10	11	15	31
%d	2	3	4	5

b) Consider the following disassembled code for function foo:

```
08048414 <foo>:
 8048414:       push   %ebp
 8048415:       mov    %esp,%ebp
 8048417:       sub    $0x18,%esp
 804841a:       cmpl   $0x20,__A__(%ebp)      0xc
 804841e:       jne    8048427 <foo+0x13>
 8048420:       mov    $0x0,%eax
 8048425:       jmp    8048446 <foo+0x32>
 8048427:       mov    __B__(%ebp),%eax       0xc
 804842a:       lea    0x1(%eax),%edx
 804842d:       mov    __C__(%ebp),%eax       0x8
 8048430:       sar    %eax
 8048432:       mov    %edx,__D__(%esp)       0x4
 8048436:       mov    %eax,__E__(%esp)       0x0
 8048439:       call   8048414 <foo>
 804843e:       mov    __F__(%ebp),%edx       0x8
 8048441:       and    $0x1,%edx
 8048444:       add    %edx,%eax
 8048446:       leave
 8048447:       ret
```

There are six blanks in the code above – labeled A-F – and all are related to displacements relative to either %esp or %ebp. Fill in the blanks with the appropriate values to make this code work correctly. You should fill in a ***hexadecimal number*** for each blank – it should ***not*** be a register specifier. These are displacements relative to the stack or frame pointers.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Answer Sheet

Name:_____

1. a.

   b.

2. a.

   b.

3. a.

   b. Fill in all blanks below

_____
A

_____
B

_____
C

_____
D

_____
E

_____
F