Pointers and Generics

(a) `swap_any(ptr1, ptr2 + 2, sizeof(int) * 3);`
(b) `swap_any(&ptr1, &ptr2, sizeof(int*));`

Assembly

(a)

```
int ham(int *burr) {
    int eliza[4];
    eliza[0] = 7;
    eliza[1] = 7;
    eliza[2] = 1;

    eliza[3] = 6 * burr[0];    // part (b)


    for (int i = 0; i < 10; i+= 3) {


        for (int j =i; j < 10; j+=2) {


            burr[i + j] = eliza[0]*eliza[1]*eliza[2]*eliza[3]; //(c)


        }

    }

    if (eliza[0] > eliza[1]) {                       // part (d)

        return 8;

    }

    if (burr[0] < burr[1] && burr[0] > burr[1]) { // part (d)

        return 9;

    }

    return 10;

}
```

(b) Explanation should mention lea then add of lea result to itself. (Optional level of detail: lea works as a 3x, and add of result to itself is 3x + 3x = 6x.) Explanation should mention that imul is slow.

Assembly

(a)

```
int ham(int aaron, char **alex)

{


    int burr = 0;


    for (int i = aaron *16; /* see part (b) */


                          i > 0; i--) {


        for (int j = 0; j < (aaron*16) + 2;


                                   j += 3) {


            alex[i][j] = 'X';


            burr += (aaron * 16);


        }


    }


    return burr;


}
```

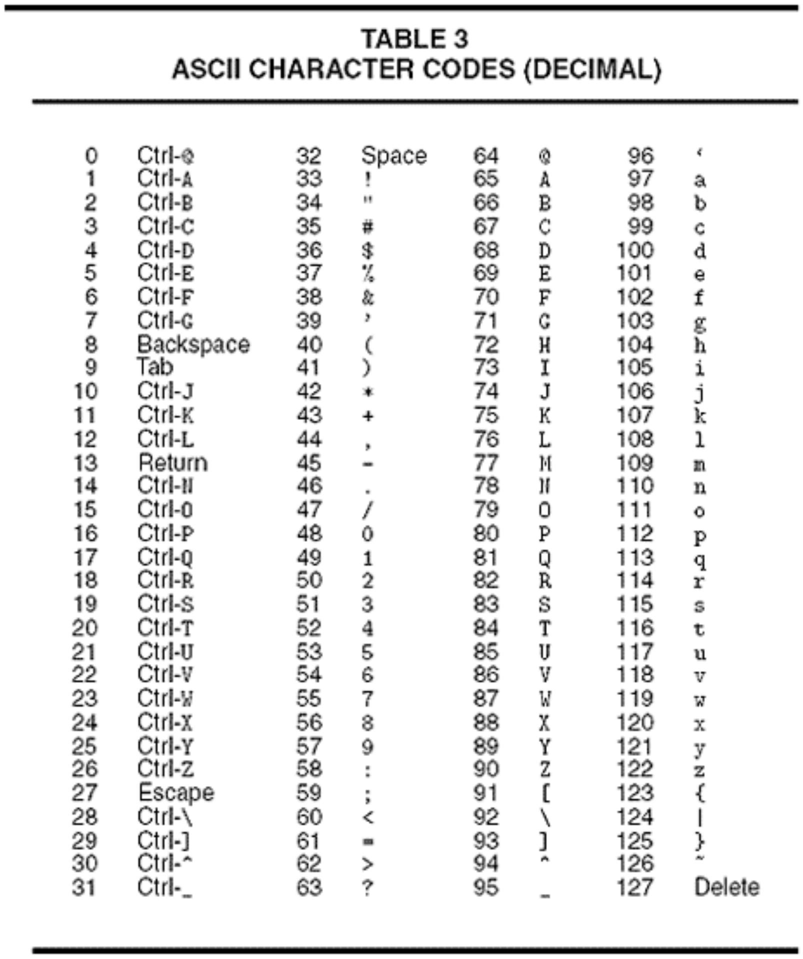

TABLE 3
ASCII CHARACTER CODES (DECIMAL)

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

(b) The shl is calculating a multiplication, because shl is faster than imul in hardware. The quantity (aaron * 16) is used in several places, so gcc does the multiplication once and then reuses the value throughout.

(c) The if has no effect because the function returns the length either way.

Assembly

(a)

```
int schuyler(int peggy)
{
    int angelica;

    int eliza = story(peggy,


                      &angelica, "helpless");


    eliza *= 2;


    return eliza + peggy;
}
```

(b)

```
int story(int raise, int *glass, char *freedom)
{
    if (freedom[0] == 'f') {

        *glass = raise;

    } else {

        *glass = 24;

    }

    int tonight = 0;


    for (int i = raise; i >= 0; i -= 2) {


        tonight += 76;

    }


    return tonight * 3;
}
```