CS 224 Lab 4
100 Points

For this lab, please remember that a `long` is 8-bytes, an `int` is 4-bytes, and a `char` is 1-byte. Additionally, the following ASCII table should be very useful.

ASCII hexadecimal set:

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 $	25 %	26 &	27 ’
28 (	29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [	5c \	5d]	5e ^	5f _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c |	7d }	7e ~	7f del

1. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of each byte is shown to the left of the byte. In this representation of memory, addresses increase as we move down in the table.

Address	
0x94373190	4b
0x94373191	28
0x94373192	44
0x94373193	72
0x94373194	42
0x94373195	00

(a) **[1 point]** What are the contents of memory location `0x94373194` in binary?

(b) **[1 point]** What are the contents of memory location `0x94373195` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x94373190`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *a);
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+2));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c+1);
```

Output: ?

(f) **[1 point]**

```
char *d = (char *)p;
printf("%c", *(d+3));
```

Output: ?

2. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of each byte is shown above the byte. In this representation of memory, addresses increase as we move left in the table.

Address	0x637	0x636	0x635	0x634	0x633	0x632	0x631	0x630
	00	41	67	21	39	54	73	49

(a) **[1 point]** What are the contents of memory location `0x632` in binary?

(b) **[1 point]** What are the contents of memory location `0x636` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x630`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *a);
```

Output:

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+2));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c+5);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *d);
```

Output: ?

3. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of the rightmost byte on each row is shown to the right of the row. In this representation of memory, addresses increase as we move to the left and up in the table.

				Address
00	21	6c	6f	`0x77ad5f40`
6f	63	20	73	`0x77ad5f3c`
69	20	73	63	`0x77ad5f38`
20	55	59	42	`0x77ad5f34`

(a) **[1 point]** What are the contents of memory location `0x77ad5f34` in binary?

(b) **[1 point]** What are the contents of memory location `0x77ad5f3e` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x77ad5f34`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *a);
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+2));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *(d+1));
```

Output: ?

4. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of the leftmost byte on each row is shown to the left of the row. In this representation of memory, addresses increase as we move to the right and down in the table.

Address								
`0x36ca22b898`	52	69	73	65	26	53	68	6f
`0x36ca22b8a0`	75	74	00	52	6f	79	61	6c
`0x36ca22b8a8`	2e	53	74	72	6f	6e	67	00

(a) **[1 point]** What are the contents of memory location `0x36ca22b89c` in binary?

(b) **[1 point]** What are the contents of memory location `0x36ca22b8aa` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x36ca22b898`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *(a+4));
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+9));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *(d+1));
```

Output: ?

5. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of each byte is shown above the byte. In this representation of memory, addresses increase as we move right in the table.

Address	0xe08	0xe09	0xe0a	0xe0b	0xe0c	0xe0d	0xe0e	0xe0f
	39	68	5f	39	46	00	3f	6a

(a) **[1 point]** What are the contents of memory location `0xe09` in binary?

(b) **[1 point]** What are the contents of memory location `0xe0e` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0xe08`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *(a+1));
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+1));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c+1);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *d);
```

Output: ?

6. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of the leftmost byte on each row is shown to the left of the row. In this representation of memory, addresses increase as we move to the right and down in the table.

Address				
`0x7e3ecd40`	24	70	3b	70
`0x7e3ecd44`	32	65	4c	6d
`0x7e3ecd48`	41	00	49	4f
`0x7e3ecd4c`	62	50	23	5d

(a) **[1 point]** What are the contents of memory location `0x7e3ecd48` in binary?

(b) **[1 point]** What are the contents of memory location `0x7e3ecd43` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x7e3ecd40`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *(a+2));
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+1));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c+3);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *d);
```

Output: ?

7. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of each byte is shown to the right of the byte. In this representation of memory, addresses increase as we move up in the table.

	Address
00	`0x4ae76341`
21	`0x4ae76340`
21	`0x4ae7633f`
69	`0x4ae7633e`
48	`0x4ae7633d`
78	`0x4ae7633c`

(a) **[1 point]** What are the contents of memory location `0x4ae7633f` in binary?

(b) **[1 point]** What are the contents of memory location `0x4ae7633d` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x4ae7633c`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *a);
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+4));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c);
```

Output: ?

(f) **[1 point]**

```
char *d = (char *)p;
printf("%c", *(d+1);
```

Output: ?

8. **[6 points]** Consider the following portion of memory. Each table cell is a memory location, which contains a single byte, with values shown in hex, and the address of the rightmost byte on each row is shown to the right of the row. In this representation of memory, addresses increase as we move to the left and up in the table.

								Address
55	71	5d	68	6f	4d	3a	56	`0x57558610`
4f	31	60	34	00	78	78	77	`0x57558608`
5a	3e	68	34	42	79	4b	34	`0x57558600`

(a) **[1 point]** What are the contents of memory location `0x5755860e` in binary?

(b) **[1 point]** What are the contents of memory location `0x57558602` in binary?

For each of the following code snippets, which include `printf` statements, write down what would be printed out. In each part, assume that `p` is a pointer of type `void *` that has the value `0x57558600`.

(c) **[1 point]**

```
int *a = (int *)p;
printf("%x", *(a+3));
```

Output: ?

(d) **[1 point]**

```
char *b = (char *)p;
printf("%c", *(b+4));
```

Output: ?

(e) **[1 point]**

```
char *c = (char *)p;
printf("%s", c+7);
```

Output: ?

(f) **[1 point]**

```
long *d = (long *)p;
printf("%lx", *d);
```

Output: ?

9. **[17 points]** Consider the following program.

```
#include <stdio.h>

#define SIZE 4

int main() {
  int a[SIZE];
  char *s_ptr = (char *)a;

  scanf("%s", s_ptr);

  for(int i = 0; i < SIZE - 1; i++){
    printf("a[%d] = %x\n", i, a[i]);
  }

  return 0;
}
```

Write down the needed input to be sent to *scanf* so that the calls to *printf* output:

```
a[0] = 73344e42
a[1] = 2c4b5761
a[2] = 3d51402a
```

Drawing memory as in previous problems is recommended.

The input is:

10. **[17 points]** Consider the following program.

```
#include <stdio.h>

#define SIZE 4

int main() {
  int a[SIZE];
  char *s_ptr = (char *)(a+1);

  scanf("%x", a);
  scanf("%s", s_ptr);
  scanf("%x", &a[3]);

  printf("%s", (char *)a);

  return 0;
}
```

Write down the needed input to be sent to *scanf* so that the call to *printf* outputs:

```
Rise+Shout+BYU!
```

Drawing memory as in previous problems is recommended.

The input is:

11. **[18 points]** Consider the following program.

```
#include <stdio.h>

#define SIZE 24

int main() {
  char str[SIZE];
  long *u_ptr = (long*)str;
  int *i_ptr = (int *)(u_ptr + 1);
  char *c_ptr = (char *)(i_ptr + 2);

  scanf("%lx %x %x %s", u_ptr, i_ptr, i_ptr + 1, c_ptr);

  printf("str = %s\n", str);
  return 0;
}
```

Write down the needed input to be sent to *scanf* so that the call to *printf* outputs

```
str = I <3 cs 224!! (*^_^*)
```

Drawing memory as in previous problems is recommended.

The input is: