Spring 2013, CS288 Test1, 6-7:30 pm, Thur, 2/21/2013, GITC 1100

Name:

The exam assumes 32-bit Linux machines. Make sure you have five pages. Do not take any page(s) with you. Any missing page(s) will result in failure in the exam. This exam is closed book close notes. Do not exchange anything during the exam. You all have the same exam. **No questions will be answered during the exam, including typos.** I don't want to give different answers to different people. If you are in doubt, briefly state your assumptions below, including typos if any.

I have read and understood all of the instructions above. On my honor, I pledge that I have not violated the provisions of the NJIT Academic Honor Code.

Signature: **Date:**

Answers for Questions 1 to 13 (3 points each)

1	2	3	4	5	6	7	8	9	10	11	12	13

1. Given f() and main() below, calling "main" would print:
   ```
   function f() { local y=$1; local z=$2; echo $x $y $z; }
   function main() { x=1; y=2; z=3; f $x $y $z; }
   ```
   a)1 1 2   b)1 1 3   c)1 2 3   d)1 3 3   e)None of the above

2. Given f() and main() below, calling "main 3 2 1" would print:
   ```
   function f() { local y=$1; local z=$2; echo $x $y $z; }
   function main() { local x=$1; y=$2; z=$3; f $x $y $z; }
   ```
   a)3 2 1   b)3 2 2   c)3 3 2   d)3 1 1   e)None of the above

3. Given lst=(1 2 3), echo $lst would print:
   a)1   b)1 2 3   c)(1 2 3)   d)1 2 3 in 3 lines   e)None of the above

4. Given lst=(1 2 3), echo ${lst[@]} would print:
   a)1   b)1 2 3   c)(1 2 3)   d)1 2 3 in 3 lines   e)None of the above

5. Given lst=(1 2 3), echo ${#lst[@]} would print:
   a)1   b)2   c)3   d)1 2 3   e)None of the above

6. Given `char *s="Go CS288!"; sizeof(s)` would return?
   a)7   b)8   c)9   d)10   e)11

7. Given `struct x { int a,b; struct x *p,*q; }; sizeof(struct x)` would return?
   a)8   b)16   c)24   d)32

8. Given `struct x { int **a,**b; struct x *p,*q; }; sizeof(struct x)` would return?
   a)16   b)24   c)32   d)40

9. Given `s='<span class="viewcount">/acct/1,696,807/name/</span>'`, which of the following statements extracts the number with commas?
   a) `expr "$s" : ".*\/\([0-9,]*\)\/"`
   b) `expr "$s" : ".*acct\/\(.*\)\/"`
   c) `expr "$s" : ".*t\/\(.*\)name.*"`
   d) `expr "$s" : ".*acct\/\([0-9a-zA-Z]*\)\/name"`

10. Suppose you want to find lines from index.html that have matching tags such as <h2>headline</h2>, *not* <h1>headline</h2>. Choose a statement that does that.
    a) `grep '<h\([0-9]\).*</h\1>' index.html`
    b) `grep '<h\1.*</h\([0-9]\)>' index.html`
    c) `grep '<h[0-9].*</h[0-9]>' index.html`
    d) `grep '<h\([0-9]\).*</h\([0-9]\)>' index.html`

11. What would you do to remove all commas from the string you obtained above?
    a)"${s/,/}"   b)"${s/,/g}"   c)"${s/,//}"   d)"${s//,/}"

12. At the command line prompt, you type "xyz 123 abc" and hit enter, where xyz is your C executable. What is argc?
    a)3   b)4   c)5   d)6   e)None of the above

13. Consider `s='class="context-data-item" data-context-item-views="16,291,016 views" data-context-item-user=\"thelonelyisland"'`
    `expr '"$s" : ".*data-context-item=\"\(.*\) views.*"'` would return

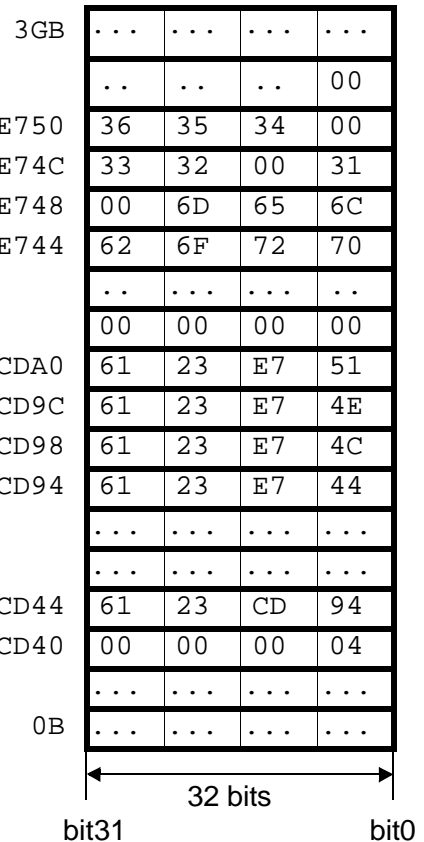   a)16,291,016   b)16291016 c)-user   d)thelonelyisland e)None of the above

14. (20 points) Write the command and parameters according to the memory map in hexadecimal shown on the right.

For example, the command that the memory map describes may be

"arguments this is a test"

Use the ASCII table below to identify what characters these hexa decimals represent. ASCII hexa 30 is character 0 (zero), hexa 70 is character p, etc.

Memory map (right side):

Address	bit31			bit0
3GB	...	...	...	...
	..	..	..	00
6123E750	36	35	34	00
6123E74C	33	32	00	31
6123E748	00	6D	65	6C
6123E744	62	6F	72	70
	..	...	...	..
	00	00	00	00
6123CDA0	61	23	E7	51
6123CD9C	61	23	E7	4E
6123CD98	61	23	E7	4C
6123CD94	61	23	E7	44
	...	...	...	...
	...	...	...	...
argv 0022CD44	61	23	CD	94
argc 0022CD40	00	00	00	04
	...	...	...	...
0B	...	...	...	...

32 bits

Decode:

- argc = 0x00000004 → 4 arguments
- argv = 0x0022CD44 points to pointer array starting at 0x6123CD94
  - argv[0] = 0x6123E744 → "problem"
  - argv[1] = 0x6123E74C → "12"
  - argv[2] = 0x6123E74E → "3"
  - argv[3] = 0x6123E751 → "456"

Command: **problem 12 3 456**

ASCII table:

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c			124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

14. Directory traversal (20 points): Write a Bash script to traverse a directory tree in *breadth*-first order using *recursion*. A seed directory is passed as a command line parameter. Return the list of *all* sub directories in complete path.

15. Structure handling (20 points): We discussed last Thursday a C program that stores video clip information using a linked list. Write a C function that adds a clip at the end of the list.

```c
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#define LEN 100

struct clip {
    int views;
    char *user, *id, *title;
    struct clip *next;
} *head;

void build_a_lst(),print_a_line(),print_lst();
void insert_at_end();
```

```c
int main(int argc, char **argv) {
    build_a_lst(*(argv+1));
    print_lst(head); /* prints all the users */
    return 0;
}
void build_a_lst(char *fn) {
    FILE *fp;
    char *line;
    if ((fp = fopen(fn,"r")) != NULL) {
        while (fgets(line, LEN, fp) != NULL) {
            insert_at_end(line);
        }
        fclose (fp);
    }}
```

/* insert a user name at the end of the linked list pointed by *head* */
/* allocate space for clip structure and user field before you can insert */
void insert_at_end(char *s) {

}