Fall 2012, CS288 Test1, 1-2:15 pm, Thur, 10/4/2012, GITC 1100

Name:

The exam assumes 32-bit Linux machines. Make sure you have all the 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. Why are you learning Linux?
   a)to broaden my horizon b)to contribute to the society
   c)my potential employer(s) would want d)to get prepared for a better future
   e)all of the above

2. 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

3. 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

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)1 2 3 c)(1 2 3) d)1 2 3 in 3 lines e)None of the above

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

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"`

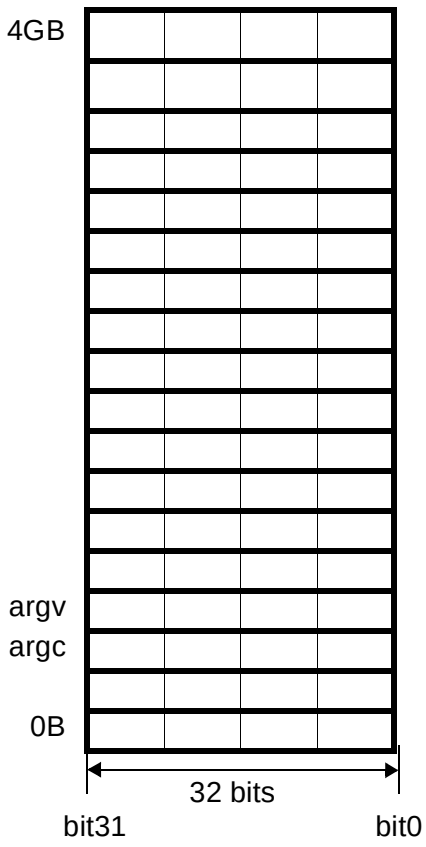
10. Suppose you want to find lines from index.html that have matching tags such as <h2>headline</h2>, *not* <h1>headline</h2>. Choose a statement that does that.
    a) `grep '<h\([0-9]\).*</h\1>' index.html`
    b) `grep '<h\1.*</h\([0-9]\)>' index.html`
    c) `grep '<h[0-9].*</h[0-9]>' index.html`
    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. Continuing on question 12, what is the total memory size to implement argc and argv in bytes?
    a)20 b)28 c)36 d)44 e)None of the above

14. (10 points) Continuing on question 13, show the contents of memory for argv, intermediate pointers, and the parameters in the table on the right. Use arrows to indicate the relationship between them.



15. Directory traversal (25 points): Write a Bash script to traverse a directory tree in *depth*-first order using *recursion.* The initial directory and depth are passed as command line parameters. Return the list of sub directories in complete path.

16. Structure handling (25 points): We discussed a C program that pushes a node to the list pointed by hp. Write a C function that creates a node with data value 4 and appends it at the end of the list. Assume build_one_two_three(); creates three nodes filled with values 1...3.

```
#include <stdio.h>
#include <malloc.h>
struct node *build_one_two_three();
struct node *append();

struct node {
int data;
struct node *next;
};
```

```
int main() {
    struct node *hp,*np; //np=new ptr
    int n;
    hp = build_one_two_three();
    // write append() function here
    // append() passes 2 parameters: hp and 4

    _____
}
```

```
//append() creates a node, attaches at the end of the list, returns the ptr to the newly created node
//use these two ptrs and the parameters passed in. do not declare/use any other ptrs or variables.

struct node *append (_____) {
    struct node *newp,*cp; //newp for new node and cp (current ptr) for list traversing









}
```