Spring 2013, CS288 Test 2, 6:00-7:15 pm, Thur, 4/4/2013, GITC 1100

Name:

The exam has 5 pages. 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. No questions will be answered during the exam. 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 to Questions 1 to 15 (2 points each) = 30 points

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Questions 1-4: For radix sorting of n signed integers on a b-bit machine, radix sort requires p passes (rounds), where n, b and p are power of 2 numbers. The integers are initially stored in lst[n] and the sorted integers will be available in lst at the end of sorting. int buf[n] is available as working space.

1. What is the number of buckets?
   a) 1 << (b/p) b)b>>p c) b/p d)b-p>>1 e) none of the above

2. The bit mask is
   a) (b>>p) - 1 b) (b-p>>1) - 1 c) (b/p) % 2 d) (1 << (b/p)) - 1
   e) none of the above

3. Assuming exactly half the integers is negative, what is the number of data assignments for the correctional step after p passes are completed? For example, moving lst[i] to buf[j], or buf[j]=lst[i]; is a considered as a data assignment.
   a) 0.5 *n* b) n c) 1.5 *n* d)2 n e) none of the above

4. For floating point radix sort, assuming exactly half the numbers is negative, what is the number of data assignments after p passes are completed for the correctional step?
   a) 0.5 *n* b) n c) 1.5 *n* d)2 n e) none of the above

5. Given `float f;` which of the C statements would allow you to access the binary equivalent of `f`:
   a) &f
   b) *f
   c) (unsigned long *) (&f)
   d) (unsigned long *) (*f)
   e) * (unsigned long *) (&f)

6. From problem 5, you now have the floating point number f converted to x. Assuming char s[32]; int i,n=32; which of the following C statements would store the binary equivalent of x in the string s, where s[0] holds the sign bit (the most significant bit) of the original number f while s[31] holds the least significant bit of the original number f:

```
a) for (i=0;i<n;i++) { s[n-1-i] = "01"[x|1]; x = x + 1; }
b) for (i=0;i<n;i++) { s[n-1-i] = "10"[x&1]; x = x + 1; }
c) for (i=0;i<n;i++) { s[n-1-i] = "01"[x&1]; x = x >> 1; }
d) for (i=0;i<n;i++) { s[n-1-i] = "10"[x&1]; x = x >> 1; }
e) none of the above
```

For 7-12 on the 15-puzzle state-space search consider succ has three nodes (p,q,r) while open has two nodes (x,y).

7. What search strategy would result in open=(p,q,r,x,y) after merging succ and open?
   a)depth b)breadth c)best d)branch-bound e)a*

8. What search strategy would result in open=(x,y,p,q,r) after merging succ and open?
   a)depth b)breadth c)best d)branch-bound e)a*

9. Depth first search relies solely on
   a)g b)h c)f=func(g,h) d)any 2 of f,g,h e)can't determine

10. Intelligent heuristic search such as A* relies on
    a)g b)h c)f=func(g,h) d)any 2 of f,g,h e)can't determine

11. What is the branching factor for the 15-Puzzle problem?
    a)2 b)3 c)4 d)5 e)can't determine

12. Assuming a 15-Puzzle state is represented as int grid[4][4], what is the amount of memory required for depth d for the average branching factor?
    a)mega's b)giga's c)tera's d)peta's e)can't determine

13. What is the basic underlying data structure of a complex web page?
    a)tree b)stack c)graph d)queue e)can't determine

14. What is an efficient way of viewing the underlying data structure of a complex web page?
    a)firebug b)excel c)notepad++ d)emacs e)can't determine

15. What do you use to convert a loose html page to xhtml?
    a)firebug b)mozilla c)libxml2 d)tagsoup e)none of the above

Problem 16 (Floating Point Radix sort - 20 points): Write a C program for sorting 32-bit floating point numbers using 8-bit (256 bins) radix sort. Use the variables listed below. Floating point numbers require a correctional step after the main loop is completed. Assume lst is initialized with n floating point numbers.

```
#define N 1048576
#define BIN 256
#define MAXBIT 32
#define LST 1
#define BUF 0

int n,group=8,bin=256;
int flag; /* to show which one holds numbers: lst or buf */
float lst[N],buf[N];
int count[BIN], map[BIN], tmap[BIN];

int main(int argc, char **argv){
    int i;
    flag = LST;
    initialize(); /* initialize lst with n random floats */
    for (i=0;i<MAXBIT;i=i+group) radix_sort(i); /* move lst to buf or buf to lst depending on the iteration number */
    correct(); /* sorted numbers must be in lst */
}
void radix_sort(int idx) {
    int i,j,k,mask; /* initialize mask for lifting the 8 least significant bits. */
    int *src_p,*dst_p; /* cast lst and buf to int pointers to treat lst/buf as int's */
    /* set src_p and dst_p*/



    /* count */



    /* map */



    /* move */



}
void correct() {




}
```

Problem 17 (splitting string - 20 points): Write a C function that splits a string *line* separated by commas and stores the values in an array of strings *fields*. The number of commas is *unknown* and your function must be able to handle any number of commas in the string. Use the following built-in functions: strtok(line, delim); strtok(NULL, delim); malloc(strlen(token)); strcpy(fields[i],token);

```
/* at the end of this function, fields will have n strings stored */
void split_line(char **fields,char *line) {
  int i=0;
  char *token, *delim;









}
```

Problem 18 (Building an array of linked lists - 30 points): Assuming you got split_line() working, complete build_lsts() that builds a list of clips. When building a list, *prepend* a clip to the list, in other words, add a clip to the front, not at the end.

```
struct clip { int number; int views; char *user; char *id; char *title; char *time; struct clip *next; };
struct clip *hourly[MAX_CLIPS];
void build_lsts(char *prefix) {
    FILE *fp; char *cmd,*filename; int i;
    for (i=0;i<MAX_CLIPS;i++) hourly[i] = NULL;
    sprintf(cmd,"ls %s*",prefix); fp = popen(cmd,"r"); i=0;
    while (fscanf(fp,"%s",filename) == 1) hourly[i++] = build_a_lst(filename); fclose(fp);
}
/* four steps: open the file, read a line at a time, call split_line() to split the line and store in fields, and call
prepend() to insert the clip to the front */
struct clip *build_a_lst(char *fn) {
    FILE *fp; struct clip *hp=NULL; char *fields[5]; char line[LINE_LENGTH];
```

```
    return hp;
}

/* three steps: malloc a clip, set values to clip BUT SET VIEWS ONLY, and add the clip to the front. */
struct clip *prepend(struct clip *hp,char **five) {
    struct clip *cp,*tp; /* tp for new clip, cp for traversing if necessary */
```

```
    return hp;
}
```