Fall 2012, CS288 Test 2, 1:00-2:15 pm, Thur, 11/15/2012, GITC 1100

Name:

The exam has 6 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.

Wlouthe @ gmail.com
sbb23 @ njit.edu
eS66@ NJ

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 10 (3 points each) = 30 points

1	2	3	4	5	6	7	8	9	10

1. The complexity of selection sort is x, merge sort is y, radix sort is z:
   a) x=O(n^2), y=O(n log n), z=O(n)
   b)x=O(n^2), y=O(n log n), z=O(n log n)
   c) x=O(n log n), y=O(n log n), z=O(n log n)
   d)x=O(n log n), y=O(n log n), z=O(n)
   e) none of the above

For problems 2 and 3, *x* and *y* may be any number including log, square, 0, and 1.

2. The complexity of merge sort is O(*x***y*) because it requires aproximately:
   a) *x* split opeartions, *y* merging opeartions
   b) *x* counting, *x* split opeartions, *y* merging comparisions
   c) *x* mapping opeartions, *x* split opeartions, *y* comparisions
   d) *x* counting, *y* merging opeartions
   e) none of the above

3. The complexity of radix sort is O(*x***y*) because it requires aproximately:
   a) *x* counting, *y* comparisions
   b) *x* mapping, *y* comparisions
   c) *x* counting, *x* mapping, *y* comparisions
   d) *x* counting, *x* mapping, *y* comparisions, *y* moving
   e) none of the above

4. Radix sort for floating point numbers requires an additional step for correction because
   a) numbers are sorted as signed integers
   b) they are biased-127 exponents
   c) negative numbers are placed in positive number places
   d) positive numbers are placed in negative number places
   e) all 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 9, 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 7-10 consider succ has three nodes (p,q,r) while open has two nodes (x,y).

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

9. Best first search relies soley on
   a)g b)h c)f=func(g,h) d)none e)any two from f,g,h

10. A* heuristic search relies on
    a)g b)h c)f=func(g,h) d)none e)any two from f,g,h

Problem 11 (Signed Integer Radix sort - 25 points): Write a C program for sorting 32-bit signed integer numbers using 8-bit (256 bins) radix sort. Use the variables listed below. You do not need any other variaables. Signed integers require an additional step after the main loop is completed. Assume lst is initialized with n signed integers.

```
#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 */
int lst[N],buf[N],count[BIN], map[BIN], tmap[BIN];
int main(int argc, char **argv){
    int i;
    flag = LST;
    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(); /* final sorted numbers must be in lst */
}}
void radix_sort(int idx) {
    int i,j,k,mask; /* set mask to lift the 8 least significant bits */
    int *src_p,*dst_p; /* use these pointers to access lst/buf */
    /* set src_p and dst_p*/



    /* count */



    /* map */



    /* move */




}

void correct() {
    int i
    for (i=0; i< N; i++)
    { if (lst[i] < 0)
        { n = i;
          break;
        }
    }
}
```

```
for (i=0; i< N; i++)
{ if ((i+n) < N)
  { buf[i] = lst[i+n] }
  else
  { buf[i] = lst[i-(N-n)]; }
}
```

Problem 12 (Sorting linked list - 25 points): In Homework4, you sorted 1440 tables on views[0], provided views[MAX_CLIP] are already sorted in descending order where views[0] has the highest number of views within each table. Given the structure listed below, write two C functions, *collect* and *rearrange*, to sort a linked list of 1440 tables in descending order of views[0]. As discussed in class, collect 1440 integers into a list and sort it. Once sorted, rearrange the linked list to keep it sorted in descendign order of views[0].

```
#define N 1440 /* tables for 24 hour period */
#define MAX_CLIP 50
struct table *hp; /* head pointer to 1440 tables */
int lst[N],buf[N],idx[N];
```

```
struct table {
    char *name[MAX_CLIP];
    char *title[MAX_CLIP];
    int views[MAX_CLIP];
    struct *prev,*next;
};
```

```
int main(int argc, char **argv){
    hp = initialize(); /* read N files and prepare a linked list of N tables */
    collect(hp); /* collect N views[0] in list[] */
    radix_sort(lst,idx); /* DO NOT write this: lst is sorted and idx holds the location in the original linked list */
    hp = rearrange(hp,lst,idx); /* actually rearrange the linked list accordign to lst and idx */
}}
/* at the end of sorting, lst is sorted in descending order while idx keeps the location of lst elements in the original linked list. DON'T DO THIS SORTING, assuming you know it from Problem 11 */
void radix_sort(int *lst, int *idx) { }

void collect(struct table *hp) { /* collect N views[0] in list[N] */
    structure table *cp; int i;
        i = 0
        cp = hp
        while (cp->next != NULL) {
            lst[i] = cp->views[0];
}           i++ }
struct table *rearrange(struct table *hp,int *lst, int *idx) {
    structure table *cp,*tp,*new_hp; /* new_hp points to the newly rearranged sorted list */
if (i == 0) {
  new_hp = (struct node *) malloc (sizeof (struct node))
  cp=new_hp
  for (j=0; j< MAX_CLIP; j++)
    { new_hp -> name[j] = tp->name[j]
      new_hp -> title[j] = tp -> title [j]
      new_hp -> views[j] = tp -> views[j] }
  new_hp -> prev = NULL;
  new_hp -> next = NULL;
} else
  { cp->next = (struct node *) malloc (sizeof (struct node))
    (cp->next) -> prev = cp
    cp = cp->next
    for (j=0; j< MAX_CLIP; j++)
      { cp->name[j] = tp ->name [j]
        cp-> title[j] = tp -> title [j]
return new_hp;
}         cp -> views[j] = tp -> views[j] }
    cp -> next = NULL
```

```
for (i=0; i<N; i++)
  tp = hp
for (j=0; j< idx[i]; j++)
  tp = tp -> next
```

Problem 13 (DOM operations - 20 points): Consider the Python code snippet bookstore.py for DOM operation, some basic methods and properties and an xml file on bookstore which we discussed this week. Write a pseudo *recursive* Python function get_text(elm) to extract all the text of a dom tree pointed by *elm*. For example, get_text(elm), where elm=<bookstore>, will return

[[u'Emacs User Manual', u'Richard Stallman', u'1980', u'12.00'],
[u'Timeline', u'Michael Chricton', u'1999', u'15.00'],
[u'Catch 22', u'Joseph Heller', u'1961', u'20.00'],
[u'Lost Symbol', u'Dan Brown', u'2009', u'15.00'],
[u'The Hitchhikers Guide to The Galaxy', u'Doug Adams', u'1978', u'10.00']]

while get_text(elm), where elm=<book category="comedy" cover="hardcopy">, will return

[u'Catch 22', u'Joseph Heller', u'1961', u'20.00']

---

bookstore.py

```
import re
from xml.dom.minidom import parse, parseString

def process_dom_tree(dm):
  lst = []
  elms = dm.getElementsByTagName('book')
  //print elms.length,elms
  for elm in elms:
     l = get_text(elm)
     lst.append(l)
  return lst

def main():
    lst = []
    global dom
    dom = parse('bookstore.xml')
    lst = process_dom_tree(dom)
    return lst

if __name__ == "__main__":
   main()


"""
methods and properties
elm.nodeType -> returns an integer # 3=text and 4=cdata
elm.data -> returns a string
elm.childNodes -> returns list
elm.attributes -> returns list
elm.getElementsByTagName(tag) -> returns list
elm.getAttribute(atr) -> returns a string
elm.hasAttributes() -> returns true (value) or false (None)
"""
```

bookstore.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="computer">
    <title lang="en">Emacs User Manual</title>
    <author>Richard Stallman</author>
    <year>1980</year>
    <price>12.00</price>
  </book>
  <book category="scifi" cover="paperback">
    <title lang="en">Timeline</title>
    <author>Michael Chricton</author>
    <year>1999</year>
    <price>15.00</price>
  </book>
  <book category="comedy" cover="hardcopy">
    <title lang="en">Catch 22</title>
    <author>Joseph Heller</author>
    <year>1961</year>
    <price>20.00</price>
  </book>
  <book category="mystery" cover="paperback">
    <title lang="en">Lost Symbol</title>
    <author>Dan Brown</author>
    <year>2009</year>
    <price>15.00</price>
  </book>
  <book category="comedy" cover="hardcopy">
    <title lang="en">The Hitchhikers Guide to The
Galaxy</title>
    <author>Doug Adams</author>
    <year>1978</year>
    <price>10.00</price>
  </book>
</bookstore>
```

Write a pseudo *recursive* Python function get_text(elm) to extract all the text of a dom tree pointed by *elm*. Pesudo being, you don't have to follow the exact Python syntax but you still have to try to stay as close as you can.

```
def get_text(elm):
    lst=[]

    return lst
```

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.

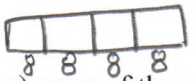
Signature:_____ **Date:**_____

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
```

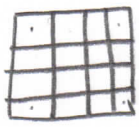
For 7-12 on the 15-puzzle state-space search consider succ has three nodes (p,q,r) while open has two nodes (x,y).

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

2 x 4 = 8
8 x 3 = 24
4 x 4 = 16
48
48 ÷ 16 = 3

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*/
    if (flag == LST){ src_p = lst; dst_p = buf; flag = BUF; }
    else { src_p = buf; dst_p = lst; flag = LST; }
    /* count */
    mask = 0xFF
    for(j=0;j<BIN;j++) { count[j]=0; map[j]=0; }
    for (i=0;i<N;i++) count[(src_p[i] >> idx) & mask]++
    /* map */
    for (j=1;j<bin;j++)
        map[j] = map[j-1] + count[j-1]
    /* move */
    for (j=0;j<N;j++)
        dest_p[map[(src_p[j]>>idx)&mask]++] = src_p[j];
}
void correct() {
    int i;
    for (i=0;i<N-n;i++)
        buf[i] = lst[N-i-1]
    for (i=0;i<n;i++)
        buf[N-n-1+i] = lst[i]
}   for (i=0;i<N;i++)
        lst[i] = buf[i];
```

```
for (i=0;i<N;i++){
    if (lst[i] < 0){
        n = i;
        break;
    }
}
```

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;
        delim= ",";
        token = strtok (line, delim)
        while (token != NULL)
        {
          field[i] = (char*) malloc (strlen (token))
          strcpy (list[i], token)
          token = strtok (NULL, delim)
          i++
        }
}
```

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;
}
```