Spring 2011, CS288 Test 3, 1:00-2:15 pm, Thur, 4/14/2011, Cul Lec 3

Name:

The exam has 5 pages with 2 problems. 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:** _____

Problem 1 (Implementing a simple dictionary - 50 points): Consider implementing a dictionary using an array of linked lists of objects, where an object consists of a word (an array of characters) and a pointer. The steps you make take are a) read a file one line at a time, b) split the line into words using space as delimiter, c) compute the hash value for each word, d) attach the word to the table. Do not be concerened about duplicates at this moment. The hash funtion is based on the ascii value of the first character of a word. For example, "CS" will be hashed to the table entry 67 while "programming" will be hashed to 112. See the example below:

INPUT FILE:

CS 288 - Intensive Programming Practicum (3-0-3) Prerequisite: CS 114
or equivalent, CS 280. The objective of this course is to raise the
level of students' programming maturity by a combination of discussion
of basic concepts and moderate practice in programming
advanced software applications. Students will use an old

HASH TABLE INDEX AND ENTRIES:

40: (3-0-3)
45: -
49: 114
50: 288 280.
67: CS CS CS
73: Intensive
80: Programming Practicum Prerequisite:
83: Students
84: The
97: a and advanced applications. an
98: by basic
99: course combination concepts
100: discussion
101: equivalent,
105: is in
108: level
109: maturity moderate
111: or objective of of of of old
112: programming practice programming
114: raise
115: students' software
116: this to the
117: use
119: will

Implement a dictionary by filling in the skeleton code listed in the next two pages.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

struct dictionary {
  int no_bins;
  struct dict_elem **htbl;
  int (*hash_fn)(void *);
  void (*add_fn)(void *, void *,int);
  //  void (*del_fn)(void *);
  //  void (*find_fn)(void *);
  void (*print_fn)(void);
};
```

```
#define NO_BINS 128
#define LINE_LENGTH 1024


struct dict_elem { void *obj, *next; };
struct dictionary *cs288_dict;
struct dictionary *init();
int hash_fn();
void add_fn(), print_fn(), build_dict();

int main(int argc, char **argv){
  cs288_dict = init(hash_fn,add_fn,print_fn);
  build_dict(argv[1]);
  cs288_dict->print_fn();
  return 0;
```

```
}
```

//fill in the formal parameters list with arguments and their types according to the call in main()

```
struct dictionary *init(_____) {
  struct dictionary *hdp;
  int i;
  hdp = malloc(sizeof(*hdp));
  hdp->no_bins = NO_BINS;
  // attach the three functions to dictionary




  hdp->htbl = malloc(NO_BINS * sizeof(struct dict_elem *));
  for (i=0; i < NO_BINS; i++) (hdp->htbl)[i] = NULL;

  return hdp;
}
```

```
int hash_fn(char *str){
  int idx;
  idx = (*str) % NO_BINS;
  return idx;
}
```

```
    /* print the table like shown above */
    void print_fn(){
      /* NOT REQUIRED. try it if time permits */
    }
```

```
void build_dict(char *intxt){
  struct dictionary *hdp = cs288_dict;
  char line[LINE_LENGTH],*delims,*tstr,*csp; /* tstr = tmp string */
  FILE *ifp;
  int idx;

  if ((ifp = fopen (intxt, "r")) != NULL) {
    delims = " \n";                    /* delimiters include space and EOL */
    while (fgets(line, LINE_LENGTH, ifp)) {
      tstr = strtok(line, delims);
      while(tstr != NULL) {
                                       csp = malloc(strlen(tstr)+1);
                                       strcpy(csp, tstr);
                                       // compute idx (table entry index) using one of the functions


      ____________________________________________________________________________________________
                                       // attach csp to the table pointed by idx, using one of the functions


      ____________________________________________________________________________________________
                                       tstr = strtok(NULL, delims);
      }
    }
    fclose (ifp);
  }
}

void add_fn(struct dictionary *dict, char *obj,int idx){
  struct dict_elem *elm,*cep;
  struct dictionary *hdp = dict;
  /* fill in to attach obj to dict entry pointed by idx */









}
```

Problem 2 (DOM operations - 50 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 in class this past week. You will write a few functions to extract values from the file. While you are not asked to write a working version of Python code, you should use the methods and properties listed below whenever appropriate.

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 e in elms:
     l = get_text(e)
     lst.append(l)
  return lst

def main():
    ml=''
    global dom
    dom = parse('books.xml')
    l = process_dom_tree(dom)
    return xml

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">Free Software Foundation</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 Gal-
axy</title>
  <author>Doub Adams</author>
  <year>1978</year>
  <price>10.00</price>
 </book>
</bookstore>
```

(a) Write a pseudo Python function to find book elements with comedy under $30.

(b) Write a pseudo Python function get_text(elm) to extract all the text of a dom tree pointed by *elm*.

(c) Write a pseudo Python function to retrieve all the text strings for the books published before 1990. Use get_text(dom).