Spring 2013 CS288 Final exam, 6 pm - 8:30 pm, Thur, 5/16/2013, GITC

Name:

The exam has 6 pages with 5 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 (splitting string - 15 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 2 (Building an array of linked lists - 20 points): Assuming you got split_line() working, complete build_lsts() that builds a list of clips. When building a list, *append* a clip at the end of the list.

```
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);
}
```

/* open the file, read a line at a time, call split_line() to split the line and store in fields, malloc a clip, set values to clip **BUT SET VIEWS ONLY** remember views is int, not char *, and append the clip at the *end* of the list */

struct clip *build_a_lst(char *fn) {

```
    FILE *fp; char *fields[5]; char line[LINE_LENGTH];
    struct clip *hp=NULL, *cp, *tp; /* tp for new clip, cp for traversing */









    return hp;
}
```

Problem 3 (Search - 20 points): In HW6, you wrote a C program that performs state-space search for the 15-Puzzle similar in principle to those used in game engines. This search consists of a main loop that operates on two major lists open and closed and a small list, successors. Write a C program that performs A* search using open, closed, succ, expand(node), filter(succ, list), merge(lst1,lst2). Write the main loop and merge(). DO NOT write filter() and expand(). Again, use the same structure used in the homework. Assume the values used for node selection are computed and insertion functions listed below are available:

```
struct node *insert_by_h(node,lst),*insert_by_g(node,lst),*insert_by_f(node,lst); /* insert a node to list */

struct node {
  int loc[N+1][N];                    /* the last row holds the three values used for node selection */
  struct node *next;
} *start,*goal;
struct node *initialize(),*expand(),*merge(),*filter();
```
int main(int argc,char **argv) {
```
      struct node *tsucc,*csucc,*copen,*topen,*open,*closed,*succ;
      open=closed=succ=NULL;
      start=initialize(argc,argv);        /* init initial and goal states */
      open=start;













}
/* you may use any and all of the insert functons listed above */
```
struct node *merge(struct node *succ,struct node *open,int flag) {
```
      struct node *csucc,*copen;





      return open;
}
```

Problem 4 (DOM operations - 30 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. Wite two functions; First, write a *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']
```

Second, write another function as described in (b) which uses the function you just wrote in part (a).

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 Gal-
axy</title>
    <author>Doug Adams</author>
    <year>1978</year>
    <price>10.00</price>
  </book>
</bookstore>
```

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

(b) Write a Python function to retrieve all the text strings for the books published before 1990.
Use get_text(elm) in part (a).

Problem 5 (Dot product for multicore machines - 15 points): Complete the SPMD (single program multiple data) program below for computing dot product on multicore machines using the collective MPI API's listed below. The steps are: compute size, scatter part of vectors x and y to other machines using MPI_Scatter, compute dot product in parallel on all machines and collect partial products using MPI_Gather, add the partial products to come up with a single final product by the master machine. No neeed to initialize the vectors.

```
    MPI_Scatter(src_lst,m,MPI_INT,dst_lst,m,MPI_INT,MASTER,world);
    MPI_Gather(src_buf,number,MPI_INT,dst_buf,number,MPI_INT,MASTER,world);
```

```
int main(int argc, char *argv[]) {
    int i,n,nprocs,pid,size,prod=0;
    int vector_x[NELMS],vector_y[NELMS],partial_prods[MAXPROCS];
    MPI_Status status;
    MPI_Comm world;
    n = atoi(argv[1]); /* number of elements */
    MPI_Init(&argc, &argv);
    world = MPI_COMM_WORLD;
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);











    MPI_Finalize();
}

int dot_product(int s,int e,int *x,int *y){
  int i,prod=0;
  for (i=s;i<e;i++) prod = prod + x[i] * y[i];
  return prod;
}
```