Fall 2013, CS288 Test 3, 1-2:15 pm, Fri, 11/22/2013, CKB204

Name:

The exam assumes 32-bit Linux machines. 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. Questions 1 to 17 are worth 3 points each.

ANSWERS

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:** _____

1. The number of iterations using breadth-first search is typically smaller than that of depth-first search for the 15-Puzzle.
   (a) True (b) False (c) Cannot Determine

2. The depth for a state-space search tree using depth-first search is typically smaller than that of breadth-first search for the 15-Puzzle.
   (a) True (b) False (c) Cannot Determine

3. The search strategy that picks using the smallest estimated remaining distance is called
   (a) depth-first (b) breadth-first (c) branch-bound (d) best-first (e)A*

4. The search strategy that combines the distance incurred and the estimated remaining distance is called
   (a) depth-first (b) breadth-first (c) branch-bound (d) best-first (e)A*

5. What is the minimum number of 32-bit words to represent a 15-puzzle state? An example 15-Puzzle state is '1 2 3 4 | 5 6 7 8 | 9 10 11 12 | 13 14 15 0,' where '|' is a row delimiter.
   (a) 2 (b)4 (c)8 (d) 16 (e) None of the above

Assuming the 15-Puzzle is implemented using singly-linked lists with last node NULL terminated, answer questions 6-9.

6. Given m=number of nodes on successor, n=number of nodes on open, p=number of nodes on closed, the number of pointer assignment statements to merge a list of successor nodes with open for *depth* first search is:
   (a) O(m) (b) O(n) (c) O(p) (d) O(n)+O(m) (e) None of the above

7. Given m=number of nodes on successor, n=number of nodes on open, p=number of nodes on closed, what is the number of pointer assignment statements to merge a list of successor nodes with open for *breadth* first search?
   (a) O(m) (b) O(n) (c) O(p) (d) O(n)+O(m) (e) None of the above

8. Given m=number of nodes on successor, n=number of nodes on open, p=number of nodes on closed, what is the number of pointer assignment statements to merge a list of successor nodes with open for *best* first search, assuming the open list is kept in sorted order of heuristic value?
   (a) O(m) (b) O(n) (c) O(p) (d) O(n)+O(m) (e) None of the above

9. Given m=number of nodes on successor, n=number of nodes on open, p=number of nodes on closed, what is the number of pointer assignment statements to merge a list of successor nodes to open for *A** first search?
   (a) O(m) (b) O(n) (c) O(p) (d) O(n)+O(m) (e) None of the above

.....
.....
.....

(I removed those irrelevant materials for test2 to avoid confusion. In Fall 2013 there were four tests, not three like this semester, including final. Test3 of Fall 2013 covered more than what's covered for test 3 this semester.)

.....
.....
.....

Problem 18 (Search - 25 points): In HW7, you wrote a C program that performs state-space search for the 15-Puzzle similar in principle to those used in game engines.

Write a filter function which returns a list of nodes that are not on the list pointed by hp. Assume nodes_same() exists, which compares two nodes and return 0 for false or 1 for true.

```
struct node {
  int loc[N+1][N];
  struct node *next;
} *start,*goal;
int flag;
struct node *initialize(),*expand(),*merge(),*filter();
```

```
void main(int argc,char **argv) {
   struct node *open,*closed,*succ,*copen;
   open=closed=succ=NULL;
   start=initialize(argc,argv);
   open=start;
   while (open) {
      copen=open;
      open=open->next;
      succ=expand(copen);
      succ=filter(open,succ);
      succ=filter(closed,succ);
      if (goal_found(succ,goal)) break;
      if (succ) open=merge(succ,open,flag);
      copen->next=closed;
      closed=copen;
}}
```

```
struct node *filter(struct node *hp,struct node *succ){
    struct node *tp
    tp = (struct node*) malloc(size of (struct node*))
    while (hp != NULL) {
            tp = succ
                    while (succ != NULL) {
                            if (nodes_same(hp, succ) == 0) {
                                    tp = hp
                                    tp = tp -> next
                            }
                    succ = succ -> next
                    }
            hp = hp -> next
    }
    succ = tp
```

```
    return succ;
}
```