Comp Sci 1MD3
Mid-Term II 2005

Dr. Jacques Carette

Name: _____

Student No.: _____

Duration : 50 minutes

- This midterm contains 15 questions on 7 pages
- This midterm will be marked out of 50. There are 60 total marks available.
- Answer the question in the space provided.
- Do not separate the pages.
- Make sure that you do not get stuck on one question; use your time wisely.

---

1. Which of the following data structures has the drawback of "creeping forward" in memory during normal use ? *[1]*

   (a) an array

   (b) a queue

   (c) a circular stack

   (d) a doubly linked list

2. Which of the following methods to resolve collisions in a hash table result in primary clustering? *[2]*

   (a) double hashing

   (b) extend buckets with a linked list

   (c) put the elements in the next available slot

   (d) put the elements in the next available slot, looking at slots $(h(k) + i^2)$ mod m

3. Given a hash table of size m where collisions resolution is done via linked list, what does the statement "this hash table currently has a load factor of 4" mean? *[2]*

   (a) only four more elements can be stored in the hash table

   (b) all the linked lists have length 4

   (c) there are 4 elements stored in the hash table

   (d) there are $4m$ elements in the table

4. In the case of a hash table where all data is stored in the table itself, explain why cells with deleted data must be marked DELETED and not UNUSED. *[3]*

5. Give the general solution of the following recurrence equation *[6]*

$$t(n) = -t(n-1) + 6 * t(n-2).$$

You may use results from the tutorials if you wish. Also, give the complete solution when given that $t(0) = 5, t(1) = 10$.

6. Show that $5 + \cos(x) \in \Theta(1)$. *[4]*

7. Define **algorithm**. Explain what each of the 2 occurences of "finite" in the definition mean. *[5]*

8. Rewrite the print1() function in the following code to print a tree of characters (stored alphabetically) in reverse order. *[3]*

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

/* use char as datatype */
typedef char datatype;

/* useful short-hand */
#define empty_tree (tree *)NULL

/* actual definition of a binary tree type */
typedef struct tree {
    datatype data;
    struct tree *left;
    struct tree *right;
} tree;

void print1(tree *t) {
    if (t!=empty_tree) {
        print1(t->left);
        printf("%c ", t->data);
        print1(t->right);
    }
}
```

9. Derive a recurrence for the running time of the *print1()* function of the previous question. *[3]*

10. Implement 3 functions to insert, delete and search for integers in a linked list of (known) bounded size N. Implement the linked list as an array of structures (see definitions below). The list is not assumed sorted; duplicate entries are allowed. *[9]*

```
#define N 100

/* definition of struct type */
typedef struct llist {
    int data;
    int next;
} llist;

static llist l[N];

/* function prototypes */
void insert(int d); /* inserts d in llist l */
void delete(int d); /* deletes first occurence of d in llist l if it exists */
int search(int d);  /* returns the index of d in llist l, -1 otherwise */
```

11. Consider the following Python program *[4]*

```
def maker(n):
    return lambda x: x*n
```

What will the output be in the following 2 cases:

```
maker(10)(8)
maker(3)('abc')
```

12. Suppose you implement a binary tree using an array, where you store the array breadth-first (like in the tutorial), and you insert elements in the tree as they are provided. The elements are to be stored in the tree in sorted order. *[4]*

    (a) How small an array could you use to insert 31 elements? In what order would the elements have to come for this to happen?

    (b) What is the worst size that might be necessary to insert 31 elements? What order corresponds to this case?

13. Describe, in english, what abc-strings will be matched by $(a|b|c)^*abacab(a|b|c)^*$. *[2]*

14. True/False: *[2]*

    (a) $xyy4yy$ matches $x?(y^{+}4?)^{*}$

    (b) $xyzxyxxyzxx$ matches $((xy)|(yx)|(zx))^{*}$

15. BONUS: Let `int **` be the type of integer matrices. Write C code which computes $C = A \times B^{t}$ (B transpose), where A and B are general matrices. Your function should use the interface defined by the prototype below. Your code should compute this recursively (no loops allowed) and using only basic arithmetic and pointer arithmetic (no array indexing allowed). Hint: you will need some auxiliary functions, where a function for a (recursive) dot product is quite useful. *[10]*

```
/* A is l x m, B is n x m, C is l x n */
void multiply(int **A, int **B, int **C, l, m, n);
```

END TEST