COMP20003 Algorithms and Data Structures
Spring 2015 **Name:** _____
Midterm **Student ID** _____
17/9/2015
Time Limit: 45 Minutes

This paper has three questions. Answer all questions on the test paper on the printed side only. You may use the facing page for your own notes if you wish, but it will not be marked unless you clearly indicate this as your wish.

There are extra blank pages at the end of the paper.

Marks are shown out of 10, and this paper counts for 10% of your final grade. You are permitted to use pencil.

This exam contains 14 pages (including this cover page) and 8 questions.

Please do not open this test until instructed to do so.

Grade Table (for teacher use only)

Question	Points	Score
1	1	
2	1	
3	1	
4	1	
5	1	
6	1	
7	1	
8	3	
Total:	10	

Computational Complexity (2 marks)

1. (1 point) Given the following functions $f(n)$ and $g(n)$, is f in $O(g(n))$ or is f in $\Theta(g(n))$, or both? If true, specify a constant c and point n_0, if false, briefly specify why.

   (a) (1/3 points) $f(n) = 2^n$, $g(n) = 2^{2n}$.

   (b) (1/3 points) $f(n) = n!$, $g(n) = 2^n$.

   (c) (1/3 points) $f(n) = \sum_{i=1}^{n} i^k$, $g(n) = n^k$.

2. (1 point) Consider $f(n) = 1 + c + c^2 + \ldots + c^n$ where c is a positive real number, and a function $g(n)$ which satisfies $f(n) \in \Theta(g(n))$:

   (a) (1/3 points) if $c < 1$, then $g(n) =?$.

   (b) (1/3 points) if $c = 1$, then $g(n) =?$.

(c) (1/3 points) if $c > 1$, then $g(n) =?$.

Searching and Sorting (5 marks)

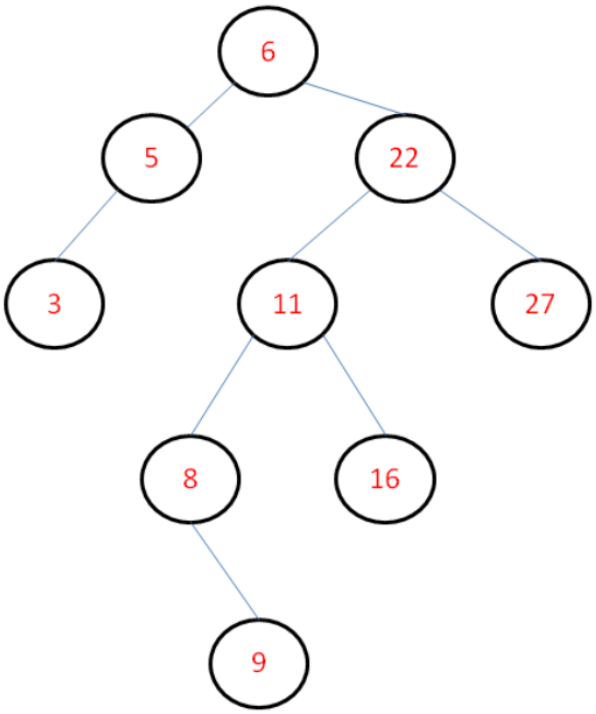
Briefly and clearly answer each of the following questions.

3. (1 point) This question is about arrays/lists and selection/insertion sort.

   (a) (1/2 points) Given m lookup/search operations over n items, is there any efficiency difference between using an unsorted linked list or an unsorted array? Under which circumstances would sorted list or array be more efficient? Answer in terms of the parameters m and n.

   (b) (1/2 points) Which method runs fastest for an array with all keys identical, selection sort or insertion sort?

4. (1 point) Which rotation do you need to get an AVL tree? Draw the resulting AVL tree



5. (1 point) Given a BST, explain how can you free all the nodes from memory.

6. (1 point) Insert the integers $\{10, 5, 4, 14, 4, 25, 23, 12, 41\}$ in a hash table of size B using linear probing and the hash function $h(x) = x\%B$, where $B = 10$

7. (1 point) Given the sequence of integers $\{ 15, 7, 3, 8, 4, 10, 2\}$, sort them using mergesort. Show the changes at every step, specify the worst case performance. Justify your answers.

C Code (3 marks)

8. (3 points) On the next page, you are given a partially written C program that reads in strings from `stdin`. The program first allocates memory to `B`, a pointer to a pointer to char, that can be used like an array.

   Fill in code in the three places indicated in the program, to:

   - Store each string in B using space efficiently.
   - Sort the strings by increasing Alphanumeric order using selection sort.
   - Free all the memory that has been used by the program, before it terminates.

   Note that the number of strings is not known at the start of the program. You can assume a maximum string length of size `MAXSTRING`.

   You are to fill in C code to accomplish this task, as indicated.

   To sort, you can use `strcmp`,

```c
int strcmp ( const char * str1, const char * str2 );
```

   which returns 0 if strings are equal, returns < 0 if $ptr1$ has lower value than $ptr2$, and > 0 viceversa.

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAXSTRING 200
#define NUMBER     10
#define OK 0

int main (argc, argv)
{
  char** B;
  char line[MAXSTRING];
  int i,j,strings,arraysize;

  if((B = (char **)malloc(NUMBER * sizeof(char *))) == NULL)
  {
    printf("malloc error\n"); fflush(stdout);
    exit(1);
  }

  strings=0;
  arraysize=NUMBER;
```

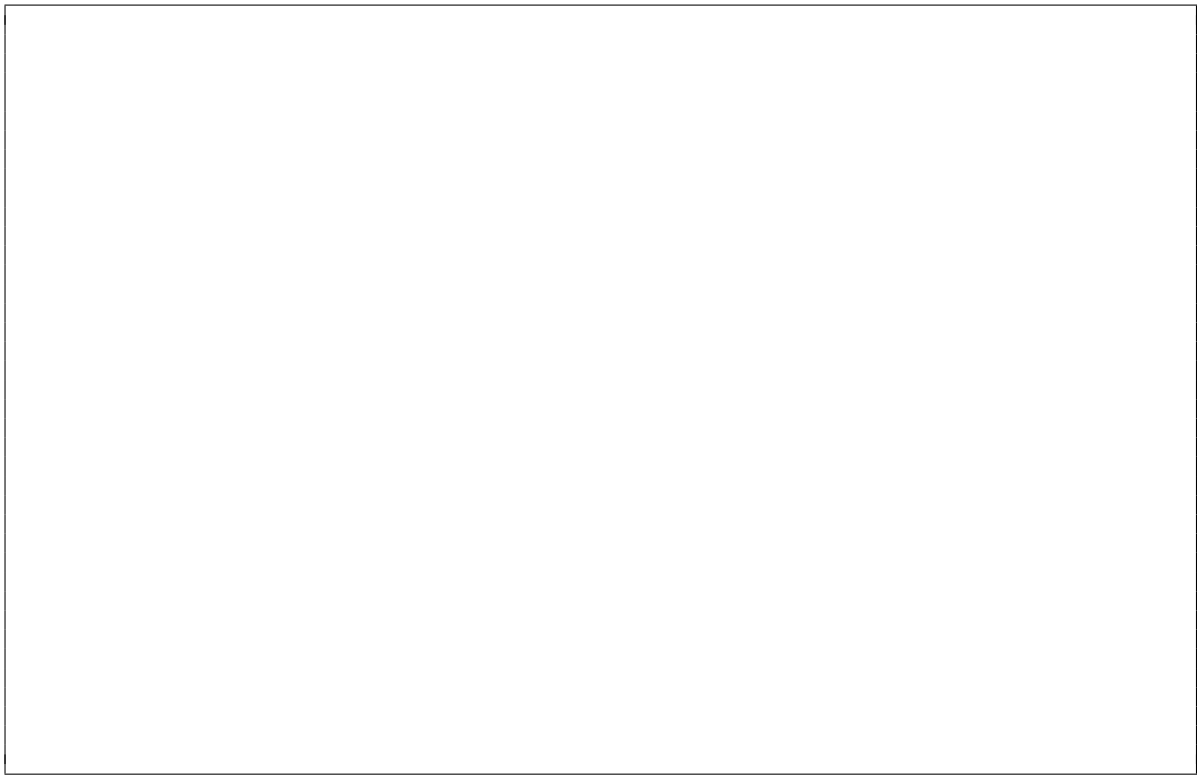
```
22
23  /**
24  * fgets reads characters from stream and stores them
25  * as a C string into str until (MAXSTRING-1) characters
26  * have been read or either a newline or the end-of-file
27  * (EOF) is reached, whichever happens first.
28  */
29  while((fgets(line,MAXSTRING,stdin))!=NULL)
30  {
31    /**
32    * ADD CODE HERE
33    * store each string, efficiently, in the array B,
34    * count the strings
35    */
```

```
}

/**
* ADD CODE HERE
* Sorting B using Selection Sort. Remember that selection sort
* repeatedly finds the smallest element in the unsorted part of
* the array and swaps it with the first element in the unsorted
* part of the array.
*/
```

```

/**
* ADD CODE HERE
* now free all the space you have
* acquired with malloc()
*/
```



```
return (OK);
}
```