A—

Calculators may be used as specified by the School as follows:

THE UNIVERSITY OF BIRMINGHAM

THIS PAGE TO BE REPLACED BY OFFICE

– —

Introduction to Computer Science

Sample Exam 2014 2 hours

[Answer all of part A and 3 out of 4 questions from part B.]

PART A: Answer all of Question 1.

1. (a) What are differences between compilers and interpreters? **[5%]**

Answer: - *Compilers include usually type checking before execution.*
- *Compilers include code optimization before execution.*
- *Compilers may lead to better performance.*
- *Interpreters may lead to better diagnostics of a programming problem.*
- *etc.*

(b) Give one advantage and one disadvantage of using an array, rather than a linked list. **[5%]**

Answer: Arrays can be indexed more efficiently, however, insertions and deletions are less efficient.

(c) Two ways of measuring the efficiency of an algorithm are: (i) measuring the time required to run it on a set of benchmark examples; (ii) analysing its time complexity. What are the key differences between these two approaches? **[6%]**

Answer: Key differences are that time complexity:

- *analyses the performance of an algorithm, independent of the programming language, compiler or hardware used to compile and run it;* **[3 marks]**
- *measures asymptotic performance (i.e. for large sizes of input, not a fixed range of sizes)* **[3 marks]**

Could also mention that time complexity usually focuses on worst-case behaviour, where as benchmarking would usually focus in average-case behaviour.

(d) Explain the difference between the notions of *partial correctness* and *total correctness* for an algorithm. **[4%]**

Answer: "Partial correctness" means that, if *the algorithm terminates, then it always produces the correct output.* **[2 marks]**
"Total correctness" means that the algorithm always terminates and *produces the correct output.* **[2 marks]**

(e) Express the following numbers in two's complement fixed point representation using 4 bits to represent the integer component and 4 bits to represent the non-integer component **[5%]**

a 3.5
b -4.25
c 7.375
d -6.514

Comment on your result for part d.

Answer:

(a) $3.5 = 0011.1000$ **[1%]**

(b) $-4.25 = -(0100.0100) = 1011.1100$ **[1%]**

(c) $7.375 = 0111.0110$ **[1%]**

(d) $-6.514 = -(0110.1010) = 1001.0110$ **[1%]**

Last one is not exact as finite bits means limited precision. **[1%]**

PART B: Answer 3 of the 4 questions in this section.

2. (a) Explain each of the following Java bytecode commands in a few words.

```
 I public int xyz(int, int, int);
0: iload_1
1: iload_2
2: imul
3: iload_3
4: iconst_5
5: iadd
6: istore 4
7: iload 4
8: ireturn
```

[10%]

Answer:

```
 I public int xyz(int, int, int);
0: iload_1      -- load an int value from a local variable 1
1: iload_2       -- load an int value from a local variable 2
2: imul           -- multiply two integers
3: iload_3           -- load an int value from local variable 1
4: iconst_5           -- load the int value 5 onto the stack
5: iadd            -- add two integers
6: istore 4 store int value into variable 4
7: iload 4 load an int value from a local variable 4
8: ireturn            -- return an integer from a method
```

(b) What is the return value of the method call `xyz(2,3,4)`? **[10%]**

Answer: 26

3. (a) What is the minimum possible depth for a binary tree containing six items?
Depth = 2

(b) Draw a balanced binary search tree containing six items with values: 1,2,3,4,5,6
Two possible answers as shown in Figures Q1 and Q2

[15%]

4. (a) For the following search tree state the order in which the nodes will be expanded assuming that the left branch of a node is expanded before the right branch.
i. depth first search **[5%]**
ii. breadth first search **[5%]**

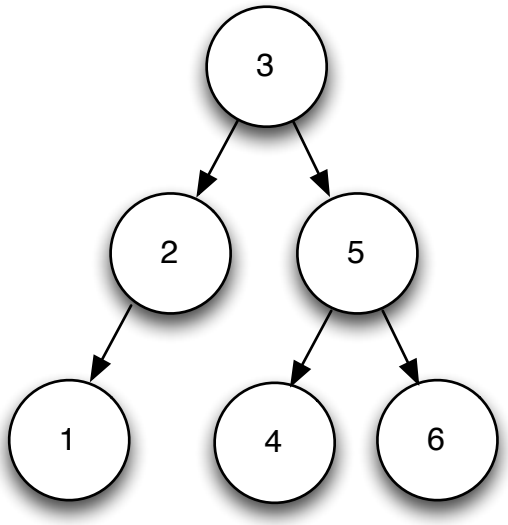

Figure Q1

(b) State an advantage of Depth First search over Breadth first search, and one disadvantage. **[10%]**

Answer: a) i) A B D E C F ii) A B C D E F b) DFS has non-exponential space complexity, whereas BFS has exponential space complexity. BFS is admissible and complete whereas DFS is neither.

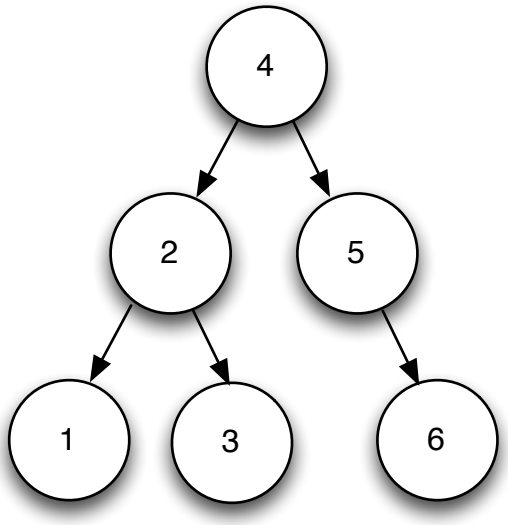

Figure Q2

5. (a) Below is some pseudo-code for an algorithm to compute the maximum distance between `n` points, stored in an array, called `points`. For any point `p`, we can access the `x` and `y` coordinates as `p.x` and `p.y`, respectively.

```
1.  maxDist = 0
2.  for i=0...n-1:
3.      for j=0...n-1:
4.          p1 = points[i]
5.          p2 = points[j]
6.          dist = sqrt((p1.x-p2.x)^2 + (p1.y-p2.y)^2)
7.          if dist > maxDist:
8.              maxDist = dist
9.  return maxDist
```

Give both the time complexity and the complexity class for this algorithm and explain your reasoning. **[10%]**

Answer: For the time complexity, we need to consider:

- *A measure of the size of the input, which we can take to be* `n`
  [2 marks]
- *What constitutes an "operation"? A reasonable assumption here would be to consider each arithmetic operation (but it would also be fine to include assignments or array look-ups).* **[2 marks]**
- *What is the worst-case behaviour of the algorithm? Here, the algorithm always loops through all* n^2 *pairs of points, regardless of their value.* **[2 marks]**

So, assuming the above, the time complexity is $T(\mathtt{n}) = 8\mathtt{n}^2$, **[2 marks]** *and the corresponding complexity class is* $O(\mathtt{n}^2)$. **[2 marks]**

(b) Assume that some additional code is appended to the end of this algorithm, which loops through the points and finds the average of the x-coordinate of the points (by summing them and diving by `n`). Explain how this affects the overall time complexity and complexity class of the algorithm. **[6%]**

Answer: The additional code would comprise a single `for` *loop, which always requires* `n` *iterations, each one performing a single addition.* **[2 marks]** *Including the additional division, the time complexity for the extra loop is* $T(\mathtt{n}) = n + 1$, *yielding a total time complexity of* $T(\mathtt{n}) = 8\mathtt{n}^2 + n + 1$. **[2 marks]** *The overall complexity class remains the same:* $O(\mathtt{n}^2)$. **[2 marks]**

(c) Explain how you would derive the complexity class for the extended algorithm in part (b), considering only the complexity class of the original algorithm and its extension, not the precise number of operations required. **[4%]**

Answer: For two algorithms composed sequentially, we can take the "maximum" of the individual complexity classes. **[2 marks]**
Here, from $O(\mathtt{n}^2)$ *and* $O(\mathtt{n})$, *we get an overall complexity class of* $O(\mathtt{n}^2)$. **[2 marks]**

6. Translate the following pieces of assembly code into high-level language. Assume that registers `$r1`, `$r2` and `$r3` are used to store variables `x`, `y` and `i` respectively.

(a)
```
     addi  $r3,$zero,1
L1:  sub   $r4,$r3,$r1
     bgez  $r4,L2
     add   $r1,$r1,$r3
     addi  $r3,$r3,1
     j L1
L2:  add  $r2,$r2,$r1
```

[7%]

(b)
```
     sub  $r4,$r3,$ri
     blez $r4,L1
     addi $r2,$r2,1
     j L3
L1:  bgez $r4,L2
     addi $r2,$r2,-1
     j L3
L2:  addi $r2,$r2,0
L3:  mult $r2,$r2,$r1
```

[8%]

You may wish to refer to the following table of instructions:

branch-on-greater-than-or-equal-to-zero	`bgez $rx,L`	`if rx>=0 go to L`
branch-on-less-than-or-equal-to-zero	`blez $rx,L`	`if rx<=0 go to L`
addition	`add $rz,$rx,$ry`	`rz=rx+ry`
immediate addition	`addi $rz,$rx,c`	`rz=rx+c`
subtraction	`sub $rz,$rx,$ry`	`rz=rx-ry`
multiplication	`mult $rz,$rx,$ry`	`rz=rx*ry`
jump	`j L`	`Go to L`

Answer:

(a)
```
for(i=1;i<x;i++){
    x+=i;
}
y+=x;
```

[7%]

(b)
```
if(x>i)
    y=1;
elseif(x<i)
    y=0;
else
```

```
    y=0;
y=y*x;
```

[8%]