Monash University

Semester One 2013 Examination Period

Faculty of Information Technology

EXAM CODES: FIT1008
TITLE OF PAPER: COMPUTER SCIENCE
EXAM DURATION: 3 hours writing time
READING TIME: 10 minutes

THIS PAPER IS FOR STUDENTS STUDYING AT:(tick where applicable)

☐ Berwick ✓ Clayton ☐ Malaysia ☐ Off Campus Learning ☐ Open Learning
☐ Caulfield ☐ Gippsland ☐ Peninsula ☐ Enhancement Studies ☐ Sth Africa
☐ Pharmacy ☐ Other (specify)

During an exam, you must not have in your possession, a book, notes, paper, calculator, pencil case, mobile phone or other material/item which has not been authorised for the exam or specifically permitted as noted below. Any material or item on your desk, chair or person will be deemed to be in your possession. You are reminded that possession of unauthorised materials in an exam is a discipline offence under Monash Statute 4.1.

No examination papers are to be removed from the room.

Please write your answers in this booklet in the spaces provided.

AUTHORISED MATERIALS

CALCULATORS	☐ **YES**	✓ **NO**
OPEN BOOK	☐ **YES**	✓ **NO**
SPECIFICALLY PERMITTED ITEMS	☐ **YES**	✓ **NO**

Candidates must complete this section if required to write answers within this paper

STUDENT ID _ _ _ _ _ _ _ _ DESK NUMBER _ _ _ _

Question		Mark
1		5
2		3
3		15
4		10
5		6
6		7
7		6

Question		Mark
8		13
9		12
10		6
Total		83

Blank page

Question 1 (5 marks)

5	

Suppose the keys on the first row of a standard keyboard (QWERTYUIOP) are inserted in succession into an initially empty binary search tree. Draw the binary search tree after each insertion.

Question 2 (3 marks)

3	

Suppose you are asked to maintain a collection of data with fixed content, i.e., you need to search for and retrieve existing items, but never have to add or delete items. Although the collection of data may be quite large, you may assume that it can fit in the computer's memory. Which of the following is the most efficient data type and data structure to use for this task: a list implemented with a sorted array, a linked list, a binary search tree, or a queue? Explain your choice.

15	

Question 3 (5+8+2=15 marks)

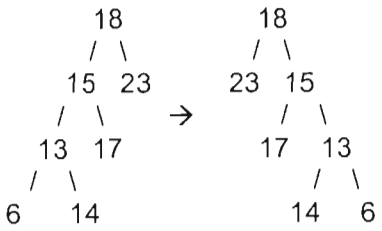
Consider a binary search tree that is constructed of nodes that are instances of the following class:

```
public class Node {
   public int key;
   public Object data;
   public Node left; // reference to the left child
   public Node right; // reference to the right child
   public Node clone(){ ... } // creates a clone of the node
   // ... all other methods
}
```

(a) The following Java method uses recursion to search for a key in the binary search tree whose root node is referred to by the parameter **`Node root`**. If it finds the key, it returns a reference to the corresponding data item, otherwise it returns **`null`**. Rewrite the **`searchTree`** method so that it uses iteration instead of recursion.

```
public static Object searchTree(Node root, int key) {
    if (root == null)
      return null;
    else if (key == root.key)
      return root.data;
    else if (key < root.key)
      return searchTree(root.left, key);
    else
      return searchTree(root.right, key);
 }
```

(b) Write a Java method with signature `public Node mirror(Node root)` that takes a reference to the root node of a binary search tree and creates a new tree (with its own nodes) that is the mirror image of the original tree. For example: if root is a reference to the root of the tree on the left below, then the return value of `public Node mirror(Node root)` would be a reference to the root of the tree on the right below. Make sure that your method does not modify the original tree. This method is easier to write with recursion, however an iterative solution is also possible.



(c) What is the worst and best time complexity of your `mirror` method in Big O notation? Explain.

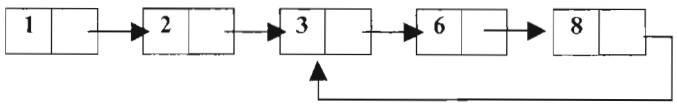
Question 4 (7+3=10 marks)

10	

Consider a **sorted linked list** constructed of nodes that are instances of the following class:

```
class Node {
   int item;
   Node next;
   // some user data and methods
}
```

Each Node in the list points to the next node, except for the last Node, which has null for next. Some lists may contain a loop; the final Node, instead of being null, has a reference to one of the previous nodes in the list. For instance, the following list contains a loop:



(a) Write a method (recursively or iteratively) with signature `boolean hasLoop(Node first)` which receives a link to the first node of a **sorted list** as argument and returns `true` if the list has a loop, and `false` otherwise.

(b) What is the worst and best time complexity in Big O notation of your method? Explain.

Question 5 (2+2+2=6 marks)

6	

This question is about object-oriented programming in Java.

(a) An abstract class cannot be declared `final`. What is the reason behind this restriction?

(b) What restrictions are placed on method overloading? (hint: think about the method signature and the return type)

(c) What restrictions are placed on method overriding? (hint: similar to part b, think about the method signature and the return type)

Question 6 (7 marks)

7	

This question is about *basic programming with queues and linked list iterators*.
Consider a `CharQueue` class, which provides, among others, the following methods:

```
public CharQueue()
public void append(char item)
```

where the data structure used to implement the queue is automatically resizable and therefore you can assume it never runs out of space. Consider also a `CharList` class, which defines a list data type of characters and provides the following methods:

```
public CharList()
public Iterator iterator()
```

where `Iterator` is a public inner class of `charList` that provides the methods:

```
public boolean hasNext()
public char next()
public void reset()
public char peek()
```

Define the method

```
public CharQueue intersection(CharList list1, CharList list2)
```

which returns a queue containing the elements that appear in both lists (`list1` and `list2`) (without duplication).

For example, `intersection(list1, list2)` for
`list1` = a, z, n, r, y
`list2` = e, a, g, z, b, n
would return the queue [a,z,n] where 2 is at the front of the queue.

For
`list1` = v, z, o, p
`list2` = o, v
the method would return the queue [v,o].

For
`list1` =
`list2` = s, t, m
it would return an empty queue [ ].

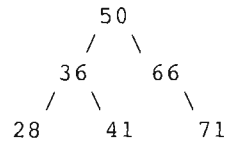
Note that at the end of the method, `list1` and `list2` must be unchanged. It is a precondition of method `intersection` that neither of the lists have duplicates. For example `list1` = a, z, a, r would be an invalid input since "a" appears more than once.

Write your answer on the blank page opposite

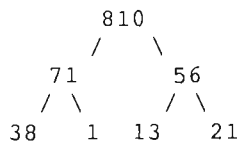
Question 7 (2+2+2=6 marks)

6	

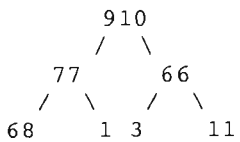
(a) State the **two different heap conditions** that have been violated, as a result of which the following binary tree cannot be called a max-heap.



(b) Draw the steps taken to insert 75 into the following max-heap.



(c) Draw the steps taken to remove one item from the following max-heap (hint: we can only remove a specific item from the heap) :



Question 8 (5+5+3=13 marks)

13	

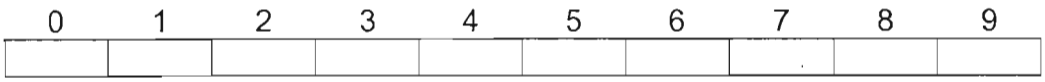
This question is about *hash tables*. Suppose you have implemented a hash table with table size **10** to store integer keys. You want to insert the following keys in the order they appear: 21, 3, 13, 7, 17, 23, 14, 12, 3, 13. The hash function used to calculate the position of each key is:

f(key) = key%10

(a) Using **separate chaining,** what will be the location in the hash table of the integer values?

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(b) Use the same hash function and give the table constructed by the linear probing method.



(c) What is the worst-case time complexity of searching an item in a hash table of N items? Think in terms of poorly chosen hash functions.

Question 9 (12 marks)

12	

Provide a *faithful* translation of the Java code into MIPS. Make sure you follow the MIPS function calling and memory usage conventions.

`double power(double x, int n){`	
`if (n == 0)`	
`return 1.0;`	

`else`
`return power(x, n - 1) * x;`
 	
`}`	

Question 10 (2+2+2=6 marks)

For each of the following, give the time complexity in Big-O notation. Explain.

6	

(a)
```
void methodA(int n) {
  int i, x=0, j;
  for (i=1; i<=n; i++)
    x=x+i;
  for (j=1; j<=x; j++)
    System.out.println("This is your last Java exam!");
}
```

(b)
```
void methodB(int n) {
  int i, j, x;
  for (i=1; i<=n; i++)
    for (j=1; j<=n+i; j++)
      System.out.println("So, all the best!");
}
```

(c)
```
void methodC(int n) {
  int i, j;
  for (i=1; i<=n; i++)
    for (j=1; j<=Math.log(i); j++)
       System.out.println("And always be proud to be a geek!");
}
```

End of exam

MIPS reference sheet for FIT1008
Semester 1, 2013

Table 1: SPIM system calls

Call code ($v0)	Service	Arguments	Returns	Notes
1	Print integer	$a0 = value to print	-	value is signed
4	Print string	$a0 = address of string to print	-	string must be terminated with '\0'
5	Input integer	-	$v0 = entered integer	value is signed
8	Input string	$a0 = address at which the string will be stored $a1 = maximum number of characters in the string	–	returns if $a1−1 characters or Enter typed, the string is terminated with '\0'
9	Allocate memory	$a0 = number of bytes	$v0 = address of first byte	-
10	Exit	-	-	ends simulation

Table 2: General-purpose registers

Number	Name	Purpose
R00	$zero	provides constant zero
R01	$at	reserved for assembler
R02, R03	$v0, $v1	system call code, return value
R04–R07	$a0--$a3	system call and function arguments
R08–R15	$t0--$t7	temporary storage (caller-saved)
R16–R23	$s0--$s7	temporary storage (callee-saved)
R24, R25	$t8, $t9	temporary storage (caller-saved)
R28	$gp	pointer to global area
R29	$sp	stack pointer
R30	$fp	frame pointer
R31	$ra	return address

Table 3: Assembler directives

.data	assemble into data segment
.text	assemble into text (code) segment
.word w1[, w2, ...]	allocate word(s) with initial value(s)
.space n	allocate n bytes of uninitialized, unaligned space
.ascii "string"	allocate ASCII string, do not terminate
.asciiz "string"	allocate ASCII string, terminate with '\0'

Table 4: Function calling convention

	Caller:	**Callee:**
On function call:	saves temporary registers on stack passes arguments on stack calls function using `jal fn_label`	saves value of $ra on stack saves value of $fp on stack copies $sp to $fp allocates local variables on stack

	Callee:	**Caller:**
On function return:	sets $v0 to return value clears local variables off stack restores saved $fp off stack restores saved $ra off stack returns to caller with `jr $ra`	clears arguments off stack restores temporary registers off stack uses return value in $v0

Table 5: MIPS instruction set

Instruction format	Meaning	Operation	Immediate	Unsigned
add Rdest, Rsrc1, Src2	Add	Rdest = Rsrc1 + Src2	addi	addu (no overflow trap)
sub Rdest, Rsrc1, Src2	Subtract	Rdest = Rsrc1 - Src2	-	subu (no overflow trap)
mult Rsrc1, Src2	Multiply	Hi:Lo = Rsrc1 * Src2	-	mulu
div Rsrc1, Src2	Divide	Lo = Rsrc1/Src2; Hi = Rsrc1 % Src2	-	divu
and Rdest, Rsrc1, Src2	Bitwise AND	Rdest = Rsrc1 & Src2	andi	-
or Rdest, Rsrc1, Src2	Bitwise OR	Rdest = Rsrc1 | Src2	ori	-
xor Rdest, Rsrc1, Src2	Bitwise XOR	Rdest = Rsrc1 $\wedge$ Src2	xori	-
nor Rdest, Rsrc1, Src2	Bitwise NOR	Rdest = (Rsrc1 | Src2)	-	-
sll Rdest, Rsrc1, Src2	Shift Left Logical	Rdest = Rsrc1 << Src2	-	-
srl Rdest, Rsrc1, Src2	Shift Right Logical	Rdest = Rsrc1 >> Src2 (MSB=0)	-	-
sra Rdest, Rsrc1, Src2	Shift Right Arithmetic	Rdest = Rsrc1 >> Src2 (MSB preserved)	-	-
mfhi Rdest	Move from Hi	Rdest = Hi	-	-
mflo Rdest	Move from Lo	Rdest = Lo	-	-
lw Rdest, Addr	Load word	Rdest = mem32[Addr]	-	-
sw Rsrc, Addr	Store word	mem32[Addr] = Rsrc	-	-
beq Rsrc1, Rsrc2, label	Branch if equal	if (Rsrc1 == Rsrc2) PC = label	-	-
bne Rsrc1, Rsrc2, label	Branch if not equal	if (Rsrc1 != Rsrc2) PC = label	-	-
slt Rdest, Rsrc1, Src2	Set if less than	if (Rsrc1 < Src2) Rdest = 1 else Rdest = 0	slti	sltu
j label	Jump	PC = label	-	-
jal label	Jump and link	$ra = PC + 4; PC = label	-	-
jr Rsrc	Jump register	PC = Rsrc	-	-
jalr Rsrc	Jump and link register	$ra = PC + 4; PC = Rsrc	-	-