

**THE UNIVERSITY OF BRITISH COLUMBIA**  
**CPSC 221: FINAL EXAMINATION**  
**\*\*SAMPLE ONLY\*\***

Name: \_\_\_\_\_ Student #: \_\_\_\_\_  
(PRINT CLEARLY)

Signature: \_\_\_\_\_

**Notes about this examination**

1. You have 2.5 hours to write this examination.
2. No notes, books, or any type of electronic equipment is allowed including cell phones and calculators.
3. There is one scratch page at the back of this exam-- *do not detach!*
4. Good luck!

**Rules Governing Formal Examinations**

1. Each candidate must be prepared to produce, upon request, a Library/AMS card for identification.
2. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.
3. No candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination.
4. Candidates suspected of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
  - a. Having at the place of writing any books, papers or memoranda, calculators, computers, audio or video cassette players or other memory aid devices, other than those authorized by the examiners.
  - b. Speaking or communicating with other candidates.
  - c. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
5. Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

Question	Mark	Max
1		6
2		2
3		2
4		8
5		8
6		6
7		6
8		5
9		4
10		4
11		6
12		8
13		4
14		2
15		2
16		8
17		4
Total		85

---

1. **{6 marks}** Give the Big-Theta complexity of each of the following, including values for  $n_0$  and  $c$ :

a.  $\frac{2n^2 + 4}{4n}$

b.  $\sqrt{n} + 5n^2$

c.  $n \lg n - 2n$

2. **{2 marks}** Consider the following C++ code involving a double-`for` loop:

```
for (int j=1; j<2*n; j=j*2)
    for (int k=n; k>n; k--)
        cout << j*k << endl;
```

What is the Big-O complexity, as a function of  $n$ ? Briefly justify your answer.

3. **{2 points}** Given the following algorithms, rewrite them below in order of *slowest* to *fastest* using the worst-case running time: {Heapify; Binary Search; Quicksort; Merge Sort}

---

**4. {8 marks}** Consider a *binary search tree* node defined as follows:

```
struct ENode {
    int      empID;
    string    empName;
    float     empSalary;
    ENode *   left;
    ENode *   right;
};
```

Complete the following function to search for the name and salary of an employee, given his/her unique employee number. For example, employee number 8675309 might be for “Sam Smith” who makes \$17.25 per hour; so, you should pass “Sam Smith” *and* 17.25 back to the caller. Also, return `true` if found; and `false` otherwise. Note that `root` is a pointer to the root of a binary search tree that is ordered by employee number (i.e., `empID`).

```
bool findEmployee(ENode * root, int key, string& name,
                  float& salary)
{
    // Return an employee's name and salary given his/her emp.ID.
    // Return: true, if found; false, if the key doesn't exist.
```

---

**5. {8 marks}** Consider a *heap* node defined as follows:

```
typedef int Key;

struct HNode {
    Key      key;
    HNode *  left;
    HNode *  right;
};
```

Write a function, `removeMin`, that given a minimum heap will remove the smallest element and return the corresponding key value, while preserving the tree as a heap. You may assume that the functions `reheapUp` and `reheapDown` are defined and available.

```
Key removeMin(HNode * root, HNode * lastParent)
// PRE: root points to the root of a minimum heap, or it's NULL
// lastParent points to the last parent in the heap
// POST: the new heap is returned (i.e. the root)
```

---

**6. {6 marks}** Hashing

Given the hash function  $h_1(k) = k \bmod 13$  and the secondary hash function  $h_2(k) = 9 - (k \bmod 7)$ , insert the following keys into the hash table in the order given:

15, 17, 30, 4, 54, 6.

If you encounter a collision, resolve it using the double hashing probe sequence you have learned in class. If an element does not fit, write it to the left of the hash table, with a note that it does not fit.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

---

7. {6 marks} Loop invariant. Consider the following function (don't worry about integer overflow):

```
int exponent ( int x, int m )
{
    int i = 1;
    int exp = x;
    while (i < m)
    {
        exp = exp * x;
        i++;
    }
    return exp;
}
```

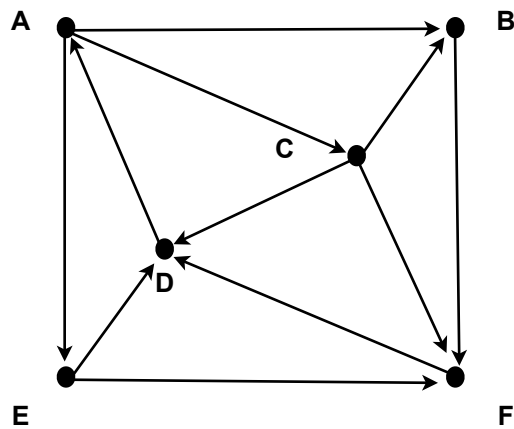
a. Identify and state the loop invariant.

$I(n)$  :

b. Use your loop invariant to prove, using mathematical induction, that the above C++ code correctly computes  $x$  to the power of  $m$ , where  $x$  and  $m$  are positive integers.

---

8. {4 marks} Consider the following directed graph:



a. Starting at vertex A, complete a depth-first traversal and write down each vertex as you *visit* it:

b. Starting at vertex F, complete a breadth-first traversal and write down each vertex as you *visit* it:

9. {4 marks} Carefully draw the graph given by the following adjacency matrix:  
Label your vertices 1 through 4.

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

---

**10. {4 marks}** In poker, a full house is a hand containing three cards of the same value  $s_1$  and two cards of the same value  $s_2$ , where  $s_1 \neq s_2$  and

$$s_1, s_2 \in \{1, 2, \dots, J, Q, K\}$$



Given a standard deck of 52 cards (13 cards from each suit), how many 5-card hands containing a full house are possible?

**11. {6 points}** Consider a drawer full of  $n$  individual socks. Assume that it's dark out and you're too lazy to turn on the light:

a. How many socks must you pull out to guarantee a matching pair if you assume that every sock has exactly one other matching sock in the drawer?

b. What if you were told there was only one matching pair of socks in the entire drawer? How many socks would you have to pull out to guarantee that you have the matching pair?

c. How many people must you have in a room before you are guaranteed to have two individuals with the same initials (assuming all people have exactly a first and last name, and are wearing socks-- just kidding, the socks don't matter)?



---

12. {8 marks} Consider the following array:

17   -8   3   2   6   1   0   4   -5

a. Show the resulting *minimum* heap, after applying the `heapify` algorithm shown in class. For part marks, show your work and circle the finished heap at the end.

b. Next, perform heap sort (*to completion*) on your heap from part (a). Fill in the following after each `reheapDown` operation is complete to show the contents of the array at that point:


---

13. **{4 marks}** Given the following postfix visitation/traversal of a math expression, draw a valid expression tree for it (similar to the examples from class):

2 3 4 \* - 4 8 2 / + +

14. **{2 marks}** For an order  $m$  B+ tree, what is the *minimum number of insertions* required for the tree to grow to 2 levels (i.e., to height 1)? (Assume that no deletions take place, and that the tree is currently empty.)

15. **{2 marks}** What is the *maximum number of key-pointer pairs* that can appear at the *leaf level* (in total, not just for one leaf page) of an order  $m$  B+ tree of height 1?

---

**16. {8 marks}** Insert the following key values one by one, in the order shown, into an initially empty B tree with order 3. Show the state of the tree whenever a split occurs. Use the method shown in class and in the lecture notes.

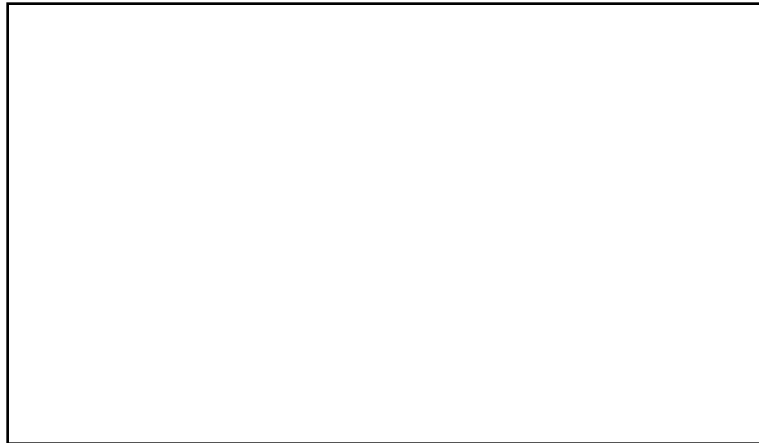
7, 2, 13, 14, 8, 21, 1, 16, 17, 22

- 
17. {4 marks} Given the following pair of C++ functions, `foo` and `bar`, show the contents of the runtime stack at its fullest (i.e. when it has the most activation records) during a call to `foo(3)`. You may assume that the stack has only the call to `main` on it at the start.

```
bool foo( int x )  
    if( x==0 ) return true;  
    else return bar( --x );
```

```
bool bar( int x )  
    if( x==0 ) return false;  
    else return foo( --x );
```

Top:



Following the principles of good coding style, if you were to rename these functions so that their names reflected what each function does, what names would you choose?

Change `foo` to...

Change `bar` to...

---

Scratch space (*do not detach*):