National University of Singapore

School of Computing

CS1020E - Data Structures and Algorithms I

(Semester 1: AY2016/17)

Date and Time: Thursday, 24 November 2016

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections.
   It comprises TWELVE (12) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** or **on free page 12** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can answer Section A in pseudo-code but you have to use C++ for Section B.
7. Write your Student Number in the box below:

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	56		
B	44		
Total	100		

A Basics (56 marks)

A.1 Interpret a C++ Code (7 marks)

You are given the following simple C++ code.

```
#include <algorithm>
#include <iostream>
using namespace std;
int main() {
  int a, b, c;
  cin >> a >> b >> c;
  if (a > b) swap(a, b);
  if (b > c) swap(b, c);
  if (a > b) swap(a, b);
  cout << a << b << c << endl;
  return 0;
}
```

Explain in English on what this C++ code is trying to do! (4 marks)

This code performs a classic algorithm that we discussed in class. Please mention its name! (3 marks)

A.2 OOP? (7 marks)

```
#include <iostream>
using namespace std;
template <typename T>
class A {
private:
  T h[1000];
public:
  A() { for (int a = 0; a < 1000; a++) h[a] = -1; }
  void set(int i, T v) { if (v >= 0) h[i] = v; }
  bool filled(int i) { return h[i] != -1; }
};
```

```
template <typename T>
class B : public A<T> {
  // yes, this is really empty
};
int main() {
  B<int> o;
  o.set(7, 77);
  o.set(0, 10);
  cout << o.filled(7) << endl;
  cout << o.filled(8) << endl;
  o.set(7, -1);
  cout << o.filled(7) << endl;
  return 0;
}
```

Explain in English on what this C++ code is trying to do! (3 marks)

Show **at least two potential issues** with this C++ code and suggest some improvements? (3 marks)

Additionally, if you have to further simplify/shorten this C++ code, what will you do? (1 mark)

A.3 Linked List (7 marks)

You are given a Singly Linked List (SLL) containing n distinct integers. The content of this SLL is already sorted in ascending order with the head being the smallest and the tail being the largest.

Now you want to implement a new operation on this sorted SLL: UPDATE(OLDVALUE, NEWVALUE) that first searches if OLDVALUE exists in the SLL. If it does not exist, then this operation does not do anything. If it exists, then this operation will update OLDVALUE to NEWVALUE and possibly reorder some ListNodes in the sorted SLL to maintain its sortedness property.

Explain in pseudo-code/English on how you will implement this operation in terms of generic SLL's operations, e.g. TRAVERSETO(INDEX), RETRIEVE(INDEX), INSERT(INDEX, VALUE), or REMOVE(INDEX) (that is, you are **not allowed** to use C++ STL list to answer this question).

Also, what will be the time complexity of your implementation?
Note: Remember the direction of the arrows in SLL.

A.4 Double-Ended Queue (7 marks)

In a normal Queue, you can only push/enqueue new item from the back of the Queue and pop/dequeue existing item from the front of the Queue. In Deque, you can push/enqueue and pop/dequeue from either the front or the back of the Deque (but you still cannot access the items in the middle).

In class, we have learned about Doubly Linked List (DLL) variant. In this question, your task is to use a DLL to implement Double-Ended Queue (Deque).

Explain in pseudo-code/English on how you will use a DLL to implement PUSH_BACK(VALUE), PUSH_FRONT(VALUE), POP_FRONT(), and POP_BACK() operations in terms of generic DLL's operations, e.g. TRAVERSETO(INDEX), INSERT(INDEX, VALUE), or REMOVE(INDEX) (that is, you are **not allowed** to use C++ STL list to answer this question).

Also, what will be the time complexity of those 4 operations in your implementation?
Note: Remember the direction**s** of the arrows in DLL.

A.5 Recursion (7 marks)

What will be the output of this C++ program (that contains an interesting recursive function f) and what is the time complexity of that recursive function f?

```
#include <iostream>
using namespace std;
int f(int n, int k) { // 2 marks for the time complexity analysis of f
  if (n == 1) return 0;
  return (f(n-1, k) + k) % n;
}
int main() { // 1 mark each for the 5 output below
  cout <<f(1,2)<<" "<<f(2,1)<<" "<<f(3,7)<<" "<<f(7,9)<<" "<<f(11,99)<< endl;
  return 0;
}
```

The output of this program are these five integers (5 marks):

The time complexity of this program is (2 marks):

A.6 Algorithm Analysis (7 marks)

What are the time complexities of the following two C++ functions?
Please give a short explanation so that we can give partial marks in case your answer is incorrect!

```
function A(int n) {
  int ans = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j += 3)
      for (int k = 0; k < 2; k++)
        ans++;
  return ans;
}
```

My time complexity analysis for function A is (3 marks):

```
function B(int n) { // assume n >= 0
  if (n <= 1) return n;
  else if (n == 2) return 1;
  else return B(n-1) + B(n-2) + B(n-3);
}
```

My time complexity analysis for function B is (4 marks):

A.7 Sorting (7 marks)

1. Let $A = [7, 15, 22, 23, 28, 30, 8, 27, 29, 35, 37, 41]$. Perform the MERGE algorithm as shown in class to merge $A[0..5]$ and $A[6..11]$ into a temporary array $B[0..11]$. Please output the resulting temporary array B after the merging!

   My temporary array B (after merging) is (1 mark):

2. Perform the PARTITION algorithm as shown in class for the following unsorted array $C = [28, 7, 30, 23, 22, 15, 41, 27, 29, 37, 8, 35]$ whereby we pick $C[0] = 28$ as the pivot of partition. Please output the resulting array C after such partition!

   My updated array C (after partitioning) is (3 marks):

3. Perform the **first two** passes of RADIX SORT algorithm as shown in class for the following unsorted array $D = [6021, 986, 2488, 9990, 2239, 5325, 5460, 1434, 9385, 9166]$ that have up to 4 digits each. Please show the content of D after we run distribute and collect operations based on the last digit and then based on the second last digit!

   My updated array D after first pass and then second pass are (3 marks):

A.8 Hashing (7 marks)

Starting from an empty hash table H with $m = 13$ cells, do the following commands one after another:

1. INSERT(7)
2. INSERT(8)
3. INSERT(20)
4. INSERT(33)
5. DELETE(20)
6. DELETE(33)
7. INSERT(59)

Please use open addressing collision resolution technique: Linear Probing to resolve collisions (if any) and show the content of hash table $H_{after-LP}$ (2 marks).

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

Then, redo the question again, but this time please use another collision resolution technique: Quadratic Probing. Show the content of hash table $H_{after-QP}$ (2 marks).

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

Finally, redo the question one last time, but this time please use yet another collision resolution technique: Double Hashing (with secondary hash function $h2(key) = 11 - key\%11$). Show the content of hash table $H_{after-DH}$ (3 marks).

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

B Applications (44 marks)

B.1 Printing Integers (14 marks)

Given three integers N ($1 \leq N \leq 10\,000\,000$), L, R, and then an integer array $A = [A_0, A_1, ..., A_{N-1}]$, your task is to print the integers in A that fall between a range $[L..R]$ (inclusive) **in sorted order and in one line**. Separate two integers in the output with a single space. All integers in array A, L, and R are positive and fit in standard 32-bit signed integer data type. You can assume that the integer values in A are distinct. Lastly, $L < R$ but $L + 10\,000 \geq R$.

For example, if $N = 10$, $L = 4$, $R = 7$, and $A = [3, 1, 9, 2, 6, 4, 5, 8, 10, 7]$, we need to output this answer: "4 5 6 7 " in one line (notice that it is OK to have a single space after 7).

Please complete the following C++ skeleton file to solve this problem **using the best possible algorithm** and write the time complexity analysis of your solution as comments.

```
#include <iostream>
#include <vector>
                                    // you can include additional library if needed
using namespace std;
int main() {                        // the input part has been given to you for free
  int i, N, L, R, v; vector<int> A;
  cin >> N >> L >> R;
  for (i = 0; i < N; i++) { cin >> v; A.push_back(v); }  // write your answer below









  return 0; //-----------------------------------------------------------------------
} /* the time complexity of this solution is O(____________)
my explanation:



*/
```

B.2 Summing Integers (17 Marks)

Given an integer array $A = [A_0, A_1, ..., A_{N-1}]$ that contains N integers ($1 \leq N \leq 20$) and a target integer K, determine if there is *at least one* **subset** of integers in A that sums to K (print "YES" or "NO" in one line according to the result). All integers in array A and K are positive and not more than 1 000. You can assume that the integer values in A are distinct.

For example, if $A = [3, 1, 4]$ (which means $N = 3$) and $K = 7$, we print 'YES' (as 3+4 sums to 7). But if A and N are the same but $K = 6$, we print 'NO' (as no subset of integers in A sums to 6).

Please complete the following C++ skeleton file whereby you have to write **any correct recursive solution** to solve this problem and write the time complexity analysis of your solution as comments.

```
#include <iostream>
#include <vector>
                                    // you can include additional library if needed
using namespace std;
// please complete this recursive function, add parameters as required
bool possible(vector<int>& A,                                                    ) {









} /* the time complexity of this recursive function is O(____________)-------------
my explanation:



*/
bool possible(vector<int>& A, int K) { // overloaded function, you have to edit---
  possible(A,                          ); // call the true recursive function above
}
int main() { // do not touch the main method
  vector<int> arr { 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};
  cout << (possible(arr, 210) ? "YES" : "NO") << endl;              // should be YES
  cout << (possible(arr, 210+1) ? "YES" : "NO") << endl;             // should be NO
  return 0;
}
```

B.3 Common Integers (13 Marks)

Given two integers N, M ($1 \leq N, M \leq 10\,000\,000$) and two integer arrays $A = [A_0, A_1, ..., A_{N-1}]$, $B = [B_0, B_1, ..., B_{M-1}]$, your task is to count and output the number of common integers between the two arrays A and B in one line. The content of array A and B are distinct but not necessarily sorted.

For example, if $N = 3$, $M = 4$, $A = [1, 2, 3]$ and $B = [1, 2, 7, 4]$, we need to output this answer: "2" in one line (as two integers: {1, 2} are common in both arrays A and B).

Please complete the following C++ skeleton file to solve this problem **using the best possible algorithm** and write the time complexity analysis of your solution as comments.

```
#include <iostream>
                                        // you can include additional library if needed
using namespace std;
int main() { // write your answer below, starting from reading N, M, and array A, B









  return 0; //-------------------------------------------------------------------------
} /* the time complexity of this solution is O(____________)
my explanation:




*/
```

– End of this Paper, All the Best; You can use this page 12 for extra writing space –