National University of Singapore

School of Computing

CS1020E - Data Structures and Algorithms I

(Semester 1: AY2016/17)

Date and Time: Saturday, 08 October 2016, 10.05-11.45 (100m)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) 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. Except Section B, you must use **C++** code for questions that demands an implementation.
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	65		
B	25		
C	10		
Total	100		

A Basics (65 marks)

A.1 Warm Up (6 marks)

To verify your understanding of a basic C++ program, please write down the output of the following code snippet and a short explanation on why you see that output in the answer box below. However, if the code does not compile or does not terminate when executed, write that fact instead of writing any output.

```
#include <iostream>
using namespace std;

int main() {
  int i;
  int iA[6];
  int X;
  int* ptr;
       X = 0;
  for (i = 0; i < 6; i++) {
    cin >> iA[i];
  }
       ptr = iA;
  for (i = 1; i < 6; i++)
        if (iA[i] == *ptr)
    {
      X++;
  }
  cout << X << endl;
  return 0;
}
/* The input given to this program is the line with 10 integers below:
7 1 2 3 4 5 6 7 8 9
*/
```

A.2 The Art of Rewriting Code (4 marks)

After understanding the C++ code presented in Question A.1, please rewrite it into another equivalent C++ code that performs *exactly the same computation* but easier to understand. The requirements:

1. Reduce code length by at least 7 lines (hence smaller box below with 'freebies' pre-written)
2. Avoid the unnecessary usage of array and pointer
3. Use better variable names
4. Use proper indentation

```
#include <iostream>
using namespace std;
int main() { // write your answer below















  return 0;
}
```

A.3 Using a Pre-Written Class (12 marks)

For Question A.3 and A.4, please refer to the following C++ class defined inside a file named as "Prime.h" that describes an list of integers from [0..U-1] and has an implemented function to decide if an integer (must be within this range) is a prime number or not.

```
#include <vector>
using namespace std;

class Prime {
protected:
  vector<bool> candidate;
  int U;
public:
  Prime(int upperbound) : U(upperbound) { // set up primes in range [2..U)
    for (int i = 0; i < U; i++) candidate.push_back(true);
    candidate[0] = candidate[1] = false;
    for (int i = 2; i < U; i++) if (candidate[i])
      for (int j = i*i; j < U; j += i) candidate[j] = false;
  }
  bool isPrime(int n) { // returns true/false if n < U is a prime/composite
    if (n < U) return candidate[n];
    else       return false; // no guarantee if n >= U
  }
};
```

Show *how to use* this class Prime in another C++ program that reads in an integer z ($4 \leq z \leq 200$) and outputs how many different pairs of *prime numbers* x and y that sums to z (x can be equal to y, e.g. $x + x = z$). However, we treat $x + y = z$ and $y + x = z$ as two *same* pairs.

Example 1: If $z = 10$, then we answer 2, as $3 + 7 = 10$ and $5 + 5 = 10$ (do not report $7 + 3 = 10$).

Example 2: If $z = 12$, then we answer 1, as $5 + 7 = 12$ (do not report $7 + 5 = 12$).

For this question, program correctness carries more weight than outright efficiency. The ‘freebies’ are already pre-written for you below.

```
#include <iostream>
#include "Prime.h"
using namespace std;
int main() { // write your answer below

  return 0;
}
```

A.4 Extending a Class (13 marks)

The Class Prime given in Question A.3 has an isPrime(n) function that works only when $n < U$. Now we want to extend the possible range of n by extending this C++ class Prime into a new subclass Prime2 and then *overrides* the method isPrime(n) so that it can works up to $n < U^2$. This way, without changing anything to your solution of Question A.3 (other than creating instance of class Prime2 instead of class Prime), we can enlarge the range of integer z that can your solution can answer from $(4 \leq z \leq 200)$ to $(4 \leq z \leq 40\,000)$.

As this is not a Mathematics class, the new and simple updated prime check algorithm is given as follows: An integer n is a prime if no smaller prime between $[2..\sqrt{n}]$ divides it. Now you already have the ability to check prime numbers between $[2..\sqrt{U^2}] = [2..U]$ in the parent class Prime. Use it to implement the updated algorithm! For this question, efficiency worth 3 marks.

```
#include "Prime.h"
class Prime2 : public Prime { // you can add additional variables if needed






public:
  // call parent's constructor so vector<bool>candidate contains primes [2..U]
  Prime2(int upperbound) : Prime(upperbound) { // you can add extra steps






  }
  // override isPrime function using the updated prime check algorithm














};
```

A.5 C++ Features - Part 1 (15 marks)

In Lecture 4 and T03, we discussed C++ STRING, ISTRINGSTREAM, and SORT method in STL algorithm. Now let's combine all these. You are given several lines of input that ends with an EOF. Each line contains lower case characters ['a'..'z'] or spaces ' '. For each line, your job is to sort *the words* of that line in alphabetical order and output the sorted words in one line. For example, if we are given:

```
line one contains five words
this is line two and is the last line
```

Your program (with the 'freebies' pre-written below) must output:

```
contains five line one words
and is is last line line the this two
```

```
#include <iostream> // hint: how to read input line by line?
#include <sstream> // hint: you probably need to use istringstream
#include <algorithm> // hint: you probably need sort routine in algorithm library
#include <vector> // hint: you probably need this ADT List
using namespace std;
int main() { // write your answer below









  return 0;
} // please use page 12 if you need more space
```

A.6 C++ Features - Part 2 (15 marks)

In Lecture 4 and T03, we also discussed various ideas of PAIR (template) class (and also triple :O). Now let's put this knowledge into practice. Given N ($1 < N < 1\,000$) in the first line and a list of N *distinct* birthdays in (D, M, Y) format in the next N lines, please output the list of those N birthdays from the youngest to the oldest person, i.e. sorted by decreasing Y ($1900 < Y < 2017$), and if ties, by decreasing M ($1 < M < 12$), and if still ties, by decreasing D ($1 < D < 31$ and will be valid day of the month). For example, if we are given a list of $N = 5$ birthdays like the list on the left, then your program (with the 'freebies' pre-written below) must output a list like the one on the right:

Input		Output
5		24 10 2011
24 5 1982		1 1 1984
23 5 1983		24 5 1983
24 10 2011	→	23 5 1983
1 1 1984		24 5 1982
24 5 1983		

```
#include <iostream> // hint: do you need to read line by line or??
#include <algorithm> // hint: you still need sort routine in algorithm library
#include <vector> // hint: you probably need this ADT List
using namespace std; // hint: if we make_pair(24, 5), make_pair(1, 1),
// make_pair(24, 10), push them into a vector, and sort it, we will have
// (1, 1), (24, 5), (24, 10), so how to use this into our advantage?
int main() { // write your answer below


















  return 0;
} // please use page 12 if you need more space
```

B Basic Linked List (25 Marks)

Linked List is an alternative data structure to implement the ADT List other than simple array or Vector. In this section, you will start from a Single Linked List (SLL) of integers showed in Figure 1 where the head/tail is pointing to a Linked List node that contains integer 1/87, respectively. Notice that the integers inside this SLL are already in sorted order. Please answer this Section B in *English*.

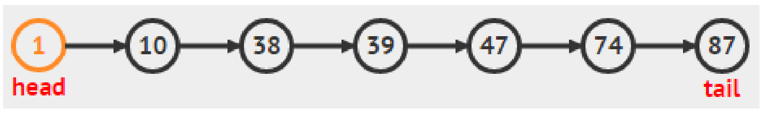

Figure 1: The starting Linked List

B.1 Insertion into a Sorted Linked List (10 Marks)

Based on your understanding on how a Linked List works, show the process to insert an integer 77 into the *sorted* SLL shown in Figure 1. The SLL *must* remains sorted after insertion. Please explain the required steps and draw the updated SLL in the space below.

B.2 Deletion from a Sorted Linked List (6 Marks)

Based on your understanding on how a Linked List works, show the process to delete an integer 77 (that you have just inserted in Question B.1) so that we re-gain the *sorted* SLL shown in Figure 1. Please explain the required steps in the space below (there is no need to draw the updated SLL as it should be the same as in Figure 1).

B.3 Printing a Sorted Linked List in Reverse Order (9 Marks)

Based on your understanding on how a Linked List works, show the process to print out the content of *the SLL shown in Figure 1* so that the output is displayed in *descending* order, i.e. output {87, 74, 47, 39, 38, 10, 1}. Of course directly printing these 7 integers without actually traversing the SLL is not a correct answer for this question. Please explain the required steps in the space below.

Section B Marks = ___ + ___ + ___ = ___

C Problem Solving (10 Marks)

Two countries A and B are at war. You are the army general of country A and has a strong belief in the effectiveness of a tight battle line formation. You have a troop of N ($1 < N < 1\,000\,000$) soldiers (indexed from 0 to N-1, initially all are alive of course) that you have put in one long and tight battle line: {0, 1, 2, ..., N-1}, where soldier 0 has no one on his left and soldier $N-1$ has no one on his right. The value N is given in the first line of the input. Then you send this troop to fight country B's army.

Over the course of the war, there are G ($0 < G \leq N$) groups of soldiers who are killed, which is described in G casualty reports. The value G is given in the second line of the input. As your troop forms a tight battle line, every time a group of soldiers is killed, their indices happen to be contiguous. That is, a casualty report is simply a pair of two integers (L, R) ($0 \leq L \leq R \leq N$-1) that describes that your soldiers with index from $\{L, L+1, ..., R-1, R\}$ are all killed simultaneously :(. As you have highlighted the importance of maintaining tight battle line, your soldiers with index $L-1$ (if exists, as L can be index 0) and $R+1$ (if exists, as R can be index N-1) must quickly 'close the gap' and march forward again. These G casualty reports are valid reports, i.e. once a soldier has perished, they will never mentioned again in future casualty reports. The G casualty reports arrive to you chronologically.

Now your job is to design a C++ program that you can use to immediately order soldier $L-1$ (if exist, or report -1 otherwise) and $R+1$ (if exist, or report -1 otherwise) to close the gap to maintain tight battle line. For example, if given $N = 10$ (soldiers) in the first line, $G = 3$ (casualty reports) in the second line, and three pairs of casualty reports: (4, 5), (0, 3), (7, 7) in the next three lines, your program must *quickly* outputs (3, 6), (-1, 6), and (6, 8), respectively in three lines, i.e.:

Input		Output
10		3 6
3		-1 6
4 5	$\rightarrow$	6 8
0 3		
7 7		

To help you understand the sample test case, here are the states of your troop for that test case:

```
0123456789 // initial state, you have 10 soldiers, with 3 casualty reports
01236789   // after 1st report (soldiers 4 and 5 perish), 3 and 6 close gap
6789       // after 2nd report (soldiers 0, 1, 2, and 3 perish), 6 is the leftmost
689        // after 3rd report (soldiers 7 perish), 6 and 8 close gap
```

Advice: As this is the last and challenging question, please do not write anything in the blank page 11 (or also page 12) until you are sure that you have answered all the other questions.

11

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