National Exams December 2019

17-Comp-A4, Program Design and Data Structures

3 Hours Duration

Notes:

1. If doubt exists as to the interpretation of a question, the candidate is urged to submit with the answer paper a clear statement of any assumptions made.
2. No calculator permitted. This is a Closed book exam.
3. Answer any six of the nine questions.
4. Any six questions constitute a complete paper. Only the first six questions as they appear in your answer book will be marked.
5. For questions that ask the candidate to write a program, **pseudocode** or any high-level language (e.g. **C** or **C++**) is acceptable unless otherwise specified. In all cases, marking will emphasize the operation of the program and not syntactic details.
6. All questions have equal weight. The total mark is out of 120.

Question 1:	(a) 10 marks; (b) 10 marks.
Question 2:	(a) 10 marks; (b) 10 marks.
Question 3:	20 marks.
Question 4:	20 marks.
Question 5:	20 marks.
Question 6:	20 marks.
Question 7:	(a) 10 marks; (b) 10 marks.
Question 8:	(a) 10 marks; (b) 10 marks.
Question 9:	20 marks.

Question 1. *Programming.*

(a) Write a program that prompts the user to input a birthdate and responds by printing the horoscope sign corresponding to the birthdate. The birthdate format is month (1-12), followed by a space, then followed by a day (1-31). The program should check for invalid months or invalid days within a month.

Here are some examples:

```
Enter birthdate: 10 18
Sign is Libra

Enter birthdate: 1 12
Sign is: Capricorn

Enter birthdate: 2 30
Invalid birthdate
```

Here are the horoscope signs and their dates:

Aries	March 21 – April 19
Taurus	April 20 – May 20
Gemini	May 21 – June 21
Cancer	June 22 – July 22
Leo	July 23 – August 22
Virgo	August 23 – September 22
Libra	September 23 – October 22
Scorpio	October 23 – November 21
Sagittarius	November 22 – December 21
Capricorn	December 22 – January 19
Aquarius	January 20 – February 18
Pisces	February 19 – March 20

(b) Horoscope signs of the same *Element* are most compatible. There are 4 Elements in astrology and 3 signs in each: FIRE (Aries, Leo and Sagittarius), EARTH (Taurus, Virgo and Capricorn), AIR (Gemini, Libra and Aquarius) and WATER (Cancer, Scorpio and Pisces). One is most compatible with a person with the same sign or the other two signs in the same Element.

Extend your program in part **(a)** to also print the other two signs a birthdate is most compatible with.

Question 2. *Programming.*

(a) Write a program that prompts the user for a line of text and then prints out the line backwards. It should repeat until the user types an empty line. Here's an example of what a session should look like, with the user's input in *italics*:

> enter a line of text: *Computer programming is fun.*
> the reversed line is: .nuf si gnimmargorp retupmoC
> enter a line of text: *CN Tower*
> the reversed line is: rewoT NC
> enter a line of text: *hello*
> the reversed line is: olleh
> enter a line of text:

Hint: you may want to read the user input into an array. Assume the user will not enter a line of text longer than 256 characters.

(b) The *saddle element* in an array is the element that is simultaneously the smallest element in its row and the largest element in its column. For example, the element `A(3,3)=4` is the saddle element in the following 6x6 array:

```
3 4 1 6 5 9
1 7 2 4 2 1
8 9 4 5 6 8
5 3 3 3 9 5
6 2 1 1 1 6
4 2 2 8 7 4
```

Write a program to input an `nxn` two-dimensional array, determine the saddle element in the array, and print the value and location of the element. Assume that there is exactly one saddle point in the array. Also assume that all integers in the array are between 1 and 99.

Question 3. *Programming.*

In a crypto-arithmetic puzzle, a mathematical equation is written using letters. Each letter can be a digit from 0 to 9 but no two letters can be the same. This is an example of such a puzzle:

SEND + MORE = MONEY

A solution to this puzzle is D=7, E=5, M=1, N=6, O=0, R=8, S=9, and Y=2.

Write a program to read a crypto-arithmetic puzzle and print a solution to it. For simplicity, assume the puzzle is in the form $x + y = z$, where x and y are each no longer than 3 letters.

Hint: read the input terms of the puzzle into character arrays and use a nested loop for each unique letter in the puzzle.

Question 4. *File I/O.*

Write a program that will compute the number of lines and the number of words in each line in a file that contains text. A line is defined as a sequence of characters of no more than 256 characters and that ends with the end of a line. A word is defined as any string of symbols that is preceded and followed by one of the following: a blank, a comma, a period, the beginning of a line, or the end of a line.

Question 5. *File I/O.*

Write a program to *merge* two sorted input files into one sorted output file. Assume each file has a number of "records", each consisting of a sequence of characters (including white spaces) terminated by a newline character, \n (assume a maximum record size of 80 characters including the terminating newline). The input files may have different numbers of records. Prompt the user for the names of the three files.

It is best that you start your answer with a short paragraph that describes your strategy for solution, the follow with the code. Include comments in your code!

Question 6. *Object-Oriented Design.*

Complex numbers occur in many engineering applications. However, most languages, including **C++** and **Java**, do not have "complex number" as a data type, nor do they directly support complex arithmetic.

Design and write a **C++** class (call it **Complex**) for supporting complex numbers and their arithmetic. Your class should allow for the declaration of complex numbers with and without initialization of real and imaginary parts. It should allow for the accessing (read & write) of the real and imaginary parts of a complex number. It should also allow for the addition, subtraction, multiplication, and printing of complex numbers.

Separate your class into a **Complex.h** header file and a **Complex.cc** implementation file.

Question 7. *Pointer-based Data Structures.*

(a) A node in a binary tree can be defined as follows, expressed in C:

```
typedef struct treenode {
    int data;
    struct element *left;
    struct element *right;
} TreeNode;
```

Write a function `preorder()` that traverses the tree using preorder traversal. The header of the function is shown below. The function must work correctly for an empty tree. Assume for each traversed node, its data is printed to the standard output.

```
/* Preorder traversal
*/
void preorder (TreeNode *root);
```

(b) Write a function **`find_min_leaf ()`** that finds the smallest data value stored in a leaf of the tree. The header of the function is:

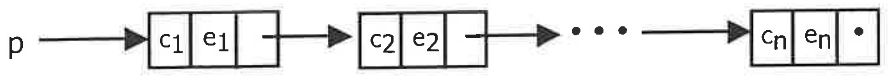
```
/* Find the smallest data value in a leaf of the
   tree pointed to by root */
int find_min_leaf (TreeNode *root);
```

Question 8. *Linked Lists.*

Consider the following definition of a **`polynomial_node`** structure expressed in C.

```
typedef struct {
    int coefficient;
    int exponent;
    polynomial_node *next;
} polynomial_node;
```

A polynomial $p(x) = c_1x^{e_1} + c_2x^{e_2} + \cdots + c_nx^{e_n}$ may be represented as a linked list of **`polynomial_nodes`**, as shown below:



where $c_1, c_2, \dots, c_n > 0$ and $e_1 > e_2 > \dots > e_n \geq 0$ are all integers.

(a) Write a function **`polynomial_node * get_polynomial()`** that reads pairs of coefficient-exponent entries **`(c,e)`**, creates a linked list representing the corresponding polynomial, and returns a pointer to the head of the newly created list. Assume the input pairs sorted by exponent values, and are terminated by a **`(0,0)`** entry.

(b) Write a function:

```
polynomial_node * add_polynomials (polynomial_ node * p1,
                                   polynomial_ node * p2)
```

that adds two the polynomials represented by the lists pointed to by **p1** and **p2** respectively. The resulting polynomial is represented by a newly created list, a pointer to which is returned by the function.

Question 9. *Algorithm Design and Sorting.*

A programmer is given an array A of random integers. The integers in the array are not in any sorted order. However, the array may contain duplicate integers. The programmer must create another array B that contains all the integers in A but without any duplicates. The integers in B need not be in any sorted order. The programmer is contemplating two ways to accomplish this task:

1. For each integer in array A, copy the integer into array B unless the item already exists in B.
2. Sort A using the Quicksort sorting algorithm. Now repeat part (a).

Given that the array contains a large number of integers, which of the above two methods will be faster? Justify your answer.