December 2016 98-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.

Marking Scheme:

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

Total mark is out of 120

**Question 1.** *Programming.*

(a) An electrical supply company manufactures electrical extension cords rated at 3.0, 6.0, 7.0, 10.0, 13.0, and 15.0 amperes. If an extension cord of length at most 50 feet is to be used with an appliance, a cord with the same amperage rating as the appliance (or higher) can be safely used. However, if the length of the cord is more than 50 but less than 100 feet, a cord of at least the next higher amperage rating is needed. A cord longer than 100 feet should not be used. Write a program to determine and print the proper extension cord selection, given the amperage of the appliance and the cord length. Your output should look something like this:

```
Enter appliance amperage: 5.8
Enter cord length in feet: 60

Use a cord rated at 7 amperes or more.
```

(b) An International Standard Book Number (ISBN) is a code of 10 characters separated by dashes such as 0-670-82162-4. An ISBN consists of four parts: a group code, a publisher code, a code that uniquely identifies the book among the publisher's offerings, and a check character. For the ISBN 0-670-82162-4, the group code is 0, which identifies the book as one from an English-speaking country. The publisher code 670 identifies the book as a Viking Press publication. The code 82162 uniquely identifies the book among the Viking Press publications (Homer: *The Odyssey*, translated by Robert Fagles). The check character is computed as follows:

1. Compute the sum of: the first digit, plus two times the second digit, plus three times the third digit, ... , plus nine times the ninth digit.
2. Compute the remainder of this sum divided by 11. If the remainder is 10, the last character is X; otherwise, the last character is the remainder.

For example, the sum for the above ISBN number is 158. The remainder is 4, which is the last character in the ISBN.

Write a program that prompts the user for an ISBN, reads the candidate ISBN from the standard input into an array, checks whether the candidate ISBN is a valid ISBN, and prints an appropriate message to the standard output.

**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:

**(b)** Write a program that reads twenty x, y coordinate pairs and sorts them:

i. In order of increasing x values.
ii. In order of decreasing y values
iii. In order of increasing distance from (0,0).

**Question 3.** *Programming.*

A problem that often arises in the design of Very Large Scale Integration (VLSI) circuits is to determine whether or not two circuit wires inadvertently intersect. Let's assume that for all practical purposes wires are either vertical or horizontal. A pair of endpoint coordinates $(x_1,y_1:x_2,y_2)$ describes each wire. Hence, (6,8:3,8) is a horizontal wire and (2,4:2,6) is a vertical one. Consequently, a simple check of the x and y coordinates of two wires can reveal if the two wires intersect.

Write a program that reads 20 pairs of coordinates and determines which wires intersect. Your program should print the coordinate pairs of intersecting wires.

**Hint**: read endpoint pairs into two arrays.

**Question 4.** *File I/O.*

Write a program to generate personalized mail. The program takes input both from an input file and from the keyboard. The input file contains the text of a letter, except that the first name of the recipient is indicated by the three characters `#N#`. The program prompts the user for a name and then writes the letter to a second file, but with the three characters replaced by the name. The three characters `#N#` may occur more than once in the file.

**Question 5.** *File I/O.*

*Run-length encoding* is a data compression technique that exploits repetition to shorten overall length. A marker and a count replace any series of identical characters. For example, the following sequence of characters in a file

```
aaaaabbccccdddefg
```

can be replaced by `\5abb\4c\3defg`. Note that the replacement of `a` and `c` did save space, while that of `d` had no effect and that `b`, `e`, `f` and `g` were not replaced since compressing them would actually waste space.

Write a program that reads characters from a file one at a time, and uses run-length encoding to write a compressed file. Make your program as efficient in time and space as possible.

**Question 6.** *Object-Oriented Design.*

Vectors are used in many engineering applications. However, some languages do not have a "vector" data type, nor do they directly support vector operations.

Design and write a C++ class (call it **`Vector`**) for supporting vectors and their operations. Your class should allow for the declaration of a vector of a given size, without initialization of its elements. Your class should allow for the accessing (read & write) of elements of a vector. It should also allow for the addition, subtraction, multiplication, and printing of vectors. These operations should produce an error message if the vectors involved in an operation are not of the same size.

Use C++ templates to support vectors of various types (only assume **`int`**, **`float`**, and **`double`** for simplicity). Overload the usual arithmetic operators to provide addition, subtraction, and multiplication of two vectors. Also overload the equality operator to allow equality comparison of two vectors.

You have freedom to select the exact syntax of some of the above operations. State any assumption you make clearly. Separate your class into a **`Vector.h`** header file and a **`Vector.cc`** implementation file.

**Question 7.** *Pointer-based Data Structures.*

Consider the following definitions of a node structure and of a queue module.

```
typedef struct {
    int data;
    node *next;
} node;


#ifndef QUEUE_H
#define QUEUE_H

static node *top = NULL;

void make_empty(void);
int is_empty(void);
void enqueue(int i);
int dequeue(void);

#end if
```

Write an implementation of the queue module using a linked list of nodes. Recall that a queue is a list of elements with insertion (enqueueing) done at one end, the rear of the queue, and deletion (dequeueing) done at the other end, the head of the queue.

**Question 8.** *Algorithm Design.*

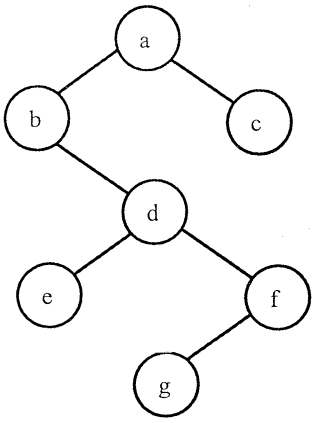
Consider an array of length n holds two different values: 0's and 1's. Write a program that puts all the 0's the left of the array and the 1's to the right. This is an example for a 15-element array:

```
Array at input:   0 1 0 1 0 0 1 1 0 1 0 1 0 0 0
Array at output:  0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
```

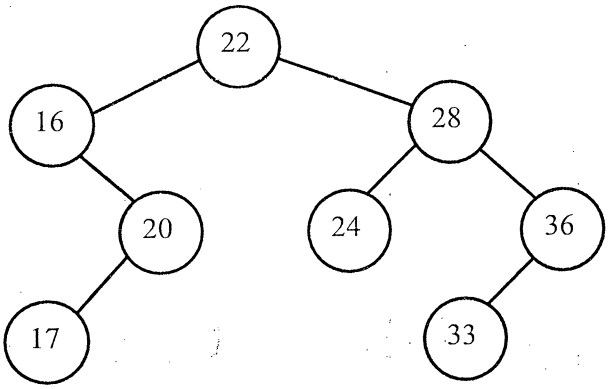
You may not use a sorting program, nor may you go through the array to count the number of 0's and 1's. You must solve the problem in one pass or traversal of the array.

**Question 9.** *Binary Trees.*

**(a)** Give the *inorder*, *preorder* and *postorder* traversals of the tree shown below.




**(b)** In a Binary Search Tree (BST), the key of each node is always smaller than the key of its right child and always greater than the key of its left child. The tree below is a BST.




Show the binary search tree after a node with key "34" is inserted.

Show the binary search tree after the node with key "33" is deleted.

Show the binary search tree after a new node with key "33" is inserted.