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. 20 marks.
3. (a) 10 marks; (b) 10 marks.
4. 20 marks.
5. (a) 5 marks; (b) 15 marks.
6. (a) 10 marks; (b) 10 marks.
7. 20 marks.
8. 20 marks.
9. 20 marks.

Total mark is out of 120.

Question 1. *Programming.*

(a) Stocks are sold and bought through a broker. The broker charges a commission that is based on the amount of a transaction according to the following sliding scale:

Transaction Amount	Commission
Under $2,500	$30 + 1.70% of transaction amount
$2,500 to $6,250	$56 + 0.66% of transaction amount
$6,250 to $20,000	$76 + 0.34% of transaction amount
$20,000 to $50,000	$100 + 0.22% of transaction amount
$50,000 to $500,000	$155 + 0.11% of transaction amount
Over $500,000	$255 + 0.09% of transaction amount

There is a minimum charge of $39.

Write a program to read the amount of a transaction and then display the amount of the commission. Here is an example:

```
Enter the amount of the transaction: 3000
Commission is: $166.00
```

(b) Airline tickets are assigned identifying numbers, such as 47715497443. To be valid, the last digit of the number must match the reminder when the other digits as a group are divided by 7. For example, 4771549744 divided by 7 yields the remainder 3, which is the last digit of 47715497443.

Write a program to read an airline ticket number and then displays whether the ticket number is valid or not.

Here is an example:

```
Enter ticket number: 47715497443
Valid
```

Here is another:

```
Enter ticket number: 47715497445
Invalid
```

Hint: read the ticket number one digit at a time.

Question 2. *Programming.*

Consider the scene in a closed-room secret design session for the 2005 model of a big-name automobile. All around the conference table are engineers and executives trying to strike a balance between greed (which leads them to build the cheapest car possible) and fear (which pulls them in the other direction, towards more reliable cars).

The relation between the mean time between failure (MTBF) of a system and its subsystems can be described as:

$$\frac{1}{MTBF_{total}} = \frac{1}{MTBF_{sub1}} + \frac{1}{MTBF_{sub2}} + \cdots + \frac{1}{MTBF_{subn}}$$

and is correct if the failure of any subsystem dooms the entire car.

Assume that the costs and MTBF's of various subsystems are:

Subsystem		MTBF	Manufacturing Cost
brakes	disc	4 years	$15.00
	drum	5 years	$25.00
engine	wankle	3 years	$1,067.00
	conventional	6 years	$1,850.00
suspension	air	9 years	$430.00
	oil	7 years	320.00
electrical	computer	2 years	130.00
	standard	4 years	$40.00

Write a program that models each combination of subsystems and answers the following questions:

1. Which system is the cheapest?
2. Which has the longest MTBF?
3. Which system has the lowest cost per failure-free year?

Hint: use nested loops.

Question 3. *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 4. *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.

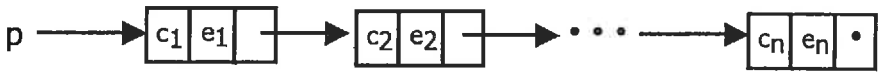
Separate your class into a **Complex.h** header file and a **Complex.cc** implementation file.

Question 5. *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 $c_1, c_2, \ldots, c_n > 0$ and $e_1 > e_2 > \ldots > e_n \geq 0$ 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.

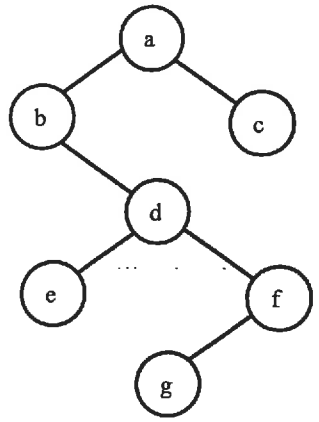
(b) Write a function:

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

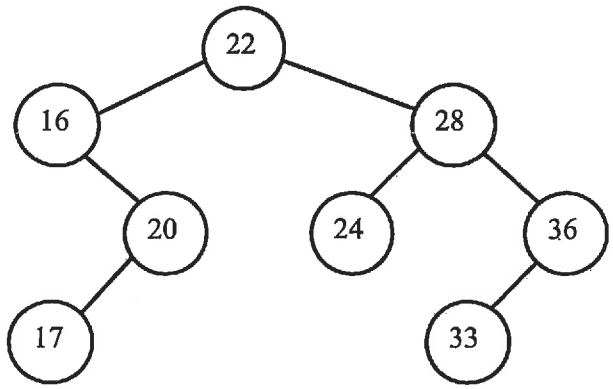
that adds two the polynomials represented by the lists pointed to by `p1` and `p2` respectively. The resulting polynomial is represented by a newly created list, a pointer to which is returned by the function.

Question 6. *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.

Question 7. *File I/O.*

Write a program to determine if two input files are "highly similar", i.e., are identical ignoring any while spaces (i.e., blanks, tabs and new lines). The input file names are to be read by your program from the standard input. Your program should return true if the files are highly similar and false otherwise. The two files may or may not be of equal sizes. Your code should return false as soon as it determines that the input files are not highly similar.

Question 8. *File I/O.*

A considerable amount of effort goes into protecting the integrity of digitized sound stored on compact disks (CDs). For example, consider the discrete "sound values" stored in the sequence:

17 15 91 68 52 84

There is essentially no protection against the loss of information. A single lost digit destroys the entire value. *Redundancy* helps minimize the danger:

17 17 17 15 15 15 91 91 91 68 68 68 52 52 52 84 84 84

Here, two complete numbers plus part of a third must be lost before an entire value is destroyed. However, since the space on which a set of replicated values are stored on the CD is very small, even the slightest damage can cause problems. To further reduce errors, the sequence is *interleaved* by spreading each triple among its neighbors:

17 15 17 91 15 17 68 91 15 52 68 91 84 52 68 84 52 84

Write a program that reads sequences of values from a file one at a time, duplicates and interleaves the values as above, and writes the coded sequence to a file. Make your program as efficient in time and space as possible. Keep in mind that the length of the sequence is variable, and is not known until the end-of-file is encountered.

Question 9. *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.