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

Total mark is out of 120, which will then be normalized to 100.
Question 1. Programming.

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:

<table>
<thead>
<tr>
<th>Transaction Amount</th>
<th>Commission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under $2,500</td>
<td>$30 + 1.70% of transaction amount</td>
</tr>
<tr>
<td>$2,500 to $6,250</td>
<td>$56 + 0.66% of transaction amount</td>
</tr>
<tr>
<td>$6,250 to $20,000</td>
<td>$76 + 0.34% of transaction amount</td>
</tr>
<tr>
<td>$20,000 to $50,000</td>
<td>$100 + 0.22% of transaction amount</td>
</tr>
<tr>
<td>$50,000 to $500,000</td>
<td>$155 + 0.11% of transaction amount</td>
</tr>
<tr>
<td>Over $500,000</td>
<td>$255 + 0.09% of transaction amount</td>
</tr>
</tbody>
</table>

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
```

Question 2. Programming.

Airline tickets are assigned identifying numbers, such as 47715497443. To be valid, the last digit of the number must match the remainder 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.

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 4. Queues.

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

```c
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);

#endif
```

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 (dequeuing) done at the other end, the head of the queue.
Question 5. *Object-Oriented Design.*

Matrices are used in many engineering applications. However, most languages, including C++ and Java, do not have a "matrix" data type, nor do they directly support matrix operations.

Design and write a C++ class (call it `Matrix`) for supporting matrices and their operations. Your class should allow for the declaration of a matrix of a given size (rows, columns), with and without initialization of its elements. Your class should allow for the accessing (read & write) of elements of a matrix. It should also allow for the addition, subtraction, multiplication, and printing of matrices.

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

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 `Matrix.h` header file and a `Matrix.cc` implementation file.

Question 6. *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 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 7. File I/O.

DNA molecules are comprised of triples that consist of four different organic bases-adenine, cytosine, guanine, and thymine, usually abbreviated A, C, G, and T respectively. Although the entire molecule can be incredibly long, a single missing base can have catastrophic effects. For example, one particular rat gene associated with hereditary diabetes differs from the normal 1000-base sequence by just one missing G residue:

normal sequence:     GGA AGC GGA GGC CGC
diabetic sequence:   GGA AGC GAC GCC GCT

The resulting frame shift mutation is apparent in the third triple; a G residue is missing, shifting subsequent residues to the left.

Write a program that is able to find the location of a frame shift mutation in a DNA base sequence. Your program should read two arbitrary long (but equal) sequences from a file; one normal and one being tested for the frame shift mutation. The length of the sequences is not known a priori; rather, a special mark indicates the end of the first sequence.

Be sure that your program confirms that the tested molecule differs by only one base, and does not diverge entirely at a point of departure.

Question 8. Pointer-based Data Structures.

(a) An element of a doubly linked list can be defined as follows, expressed in C:

```c
typedef struct element {
    int data;
    struct element *prev;
    struct element *next;
} ELEMENT;
```

Write two functions dlinked_add() and dlinked_del() that add and delete nodes from a sorted doubly linked list. The header of each function is shown below. The two functions must work correctly for empty lists.

```c
/* Insert a new node onto the list pointed to by head, keeping the list sorted */
void *dlinked_add(ELEMENT *head, ELEMENT *new);

/* Delete the node with the corresponding data value from the list pointed to by head, keeping the list sorted. If there is more than one node with the same data value delete only one. If no node exists with the data value, return NULL */
ELEMENT *dlinked_del(ELEMENT *head, int data);
```
(b) A node in a binary tree can be defined as follows, expressed in C: 

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

Write a function `find_max_leaf()` that finds the largest data value stored in a leaf of the tree. The header of the function is:

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


This is a problem about a ship caught in a terrible storm. Although there were thirty passengers on board, plus the captain, there was only enough room in the lifeboats for fifteen of them. As the captain was reluctant to leave anyone behind, she resolved to throw half of the passengers overboard before loading the boats.

As it happens, half of the passengers had slighted the captain by not dining at her table during the cruise. The captain, in revenge, arranged all the passengers in a circle and began to count. Every \( n \)'th passenger went overboard; naturally, the captain's friends were never chosen. Here is how the passengers were arranged (the captain's friends are shown with 0 and the enemies with 1):

```
0 0 0 0 1 1 1 1 0 0 1 1 0
1 1 0 0 1 1 1 0 1 0 1 0 1
```

Write a program to determine what the number \( n \) was. Start counting at the upper-left 0. A passenger is thrown overboard as soon as selected.

Assume that the arrangement of the passengers is represented by an array; each element is an integer. Use additional arrays if you wish.

**Caution:** Your solution should be really short (10-15 lines or so, excluding I/O). Excessively long solutions will be penalized.