National Exams May 2012

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.

1. All questions have equal weight.

Question 1. *Programming.*

(a) *Buoyancy* is the ability of an object to float. Archimedes' principle states that the buoyant force is equal to the weight of the fluid that is displaced by the submerged object. The buoyant force can be computed by: $F_b = V \times \gamma$

Where F_b is the buoyant force, V is the volume of the submerged object, and γ is the specific weight of the fluid. If F_b is greater than or equal to the weight of the object, then it will float, otherwise it will sink.

Write a program that inputs the weight (in pounds) and radius (in feet) of a sphere and outputs whether the sphere will sink or float in water. Use γ =62.4lb/ft^3 as the specific weight of water. The volume of a sphere is computed by $(4/3)\pi r^3$.

(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.*

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 8 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 3. *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.

Assume only matrices of type `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 4. *File I/O.*

Write a program that will compute the average word length (i.e., the average number of letters in a word) for a file that contains text. 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.*

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. *Stacks.*

Consider the following definitions of a `node` structure and of a `stack` module.

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


#ifndef STACK_H
#define STACK_H

static node *top = NULL;

void make_empty(void);
int is_empty(void);
void push(int i);
int pop(void);

#end if
```

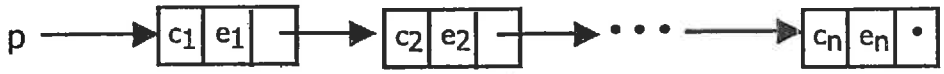
Write an implementation of the `stack` module using a linked list of `node`s. Recall that a stack is a list of elements with insertions and deletion done at only one end of the list, called the "top" of the stack.

Question 7. *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 8. *Searching.*

Given a collection of n items, each with a unique key, there exists many ways of organizing these items in order to *search* for one with a key k; i.e., determine if there exists an item in the collection whose key is equal to k. For example, it is possible to store these items in an array in no particular order (unsorted array organization). The items are searched one at a time comparing their respective keys to k until a match is found, or it is declared that no item is found.

Suggest 4 different ways of organizing the items and give the time complexity for the search as a function of n. You may assume the items are stored in memory. Which of these methods would you use if you had to search through 10 elements? Which would you use if you had to search through 3 million elements? Justify your answer in each case.

Question 9. *Algorithm Design.*

This is 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 0 1 1 1 1 0 0 1 1 0
1                             1
  1 0 0 1 1 0 1 1 0 1 0 1 0 0
```

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.