American University of Beirut
Department of Electrical and Computer Engineering

EECE 230 – Computers and Programming

Final Exam January 17, 2006

Include your name and ID as a comment in every program you write

Duration: 2 ½ Hours

(25%)

1. For faster sorting of letters, the post office encourages companies to use a bar code denoting the 5-digit zip code. For example, the zip code 95014 is represented by the bar code:

|:|::|:|:|::|:::|:::||:|::|:::|| such that

|:|::	:|:|:	||:::	:::||	:|::|	:::||
Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Check Digit

The five encoded digits are followed by a check digit, which is computed as follows: Add up all digits, and choose the check digit to make the sum a multiple of 10. For example, the zip code 95014 has a sum of 19, so the check digit is 1 to make the sum equal to 20. Each digit of the zip code, and the check digit, is encoded according to the following table where 0 denotes a column (:) and 1 denotes a vertical bar (|).

	7	4	2	1	0
0	1	1	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0

The digit can be easily computed from the bar code using the column weights 7, 4, 2, 1, 0. For example, 01100 is 0x7 + 1x4 + 1x2 + 0x1 + 0x0 = 6. The only exception is 0 which yields 11 according to the weight formula.
Write a program and store the coding table in a 2-dimensional array. Prompt the user to enter a 5-digit zip code and produce on screen the corresponding bar code. Then

prompt the user to enter a 5-digit binary combination and produce on the screen the corresponding digit. A sample run of the program would look like:

Enter zip code: 92015
The bar code is: |:|::::|:|||::::::||:|:|:::||:

Enter a sequence for a digit: 10001
The digit is: 7

(20%)

2. Design a base class "Worker" that has in the private section a string member "name" representing the first name of the worker, and a double member "rate" representing the hourly rate of the worker. In the public section you have a parameterized default constructor defaulting name to "" and rate to 0. A function "getName", a function "print" that prints the name followed by the hourly rate.

   Design two derived classes "HourlyWorker" and "SalariedWorker". Write a virtual function "computePay" that take as a parameter an integer representing the number of hours worked and that returns a double which is the corresponding pay of the worker. An hourly worker gets paid the rate times the number of hours worked if it is less than 40. For the hours more than 40, he gets 1.5 times the rate. The salaried worker gets always 40 times the rate regardless of the number of hours worked.

   Write a function "displayPay" that takes as arguments an object of type Worker and an integer for the number of hours worked. The function prints the name and rate of the worker, and then it calls the "computePay" function.

   Write a program that declares an hourly worker with name "John" and a rate of 15.5, and a salaried worker with the name "Steve" and a rate of 25.

   The output should look like this:

   Name: John Rate: 15.5

   Pay = $ 852.5

   Name: Steve Rate: 25

   Pay = $ 1000

(20%)

3. Design a simple class Square that has only one data member in the public interface representing the side of the square.

   Write a function template called "equal" that compares two parameters and return true if they are equal, and otherwise it return false.

   Write a main program to test the function equal with different data types by producing on the screen True when they are equal and False when they are not. Declare sq1 with

a side of 10, sq2 with a side 20 and sq3 with a size of 20. If you test your program with the following pairs (5, 6); (5, 5); (15.5, 15.5); (hi, hello); (sq1, sq2); (sq2, sq3) you should get on the screen on separate lines: False True True False False True.

Note: For the function to work with objects of type Square you have to overload the equality operator for this class.

(35%)

4. Design a class "Person" that has in the private section three fields. The first "name" is a string representing the first name of the person. The second "best" is a pointer to type Person representing the person who is the best friend of this person. The third "popularity" is an integer indicating how popular this person is. In other words, how many other persons have him as a best friend.

The public interface should have the functions: a parameterized default constructor, with 1 parameter, the name of the person defaulting to "", and the constructor will set best to NULL and popularity to 0. The function "increment" which increments the value of the popularity by 1. The function "getName", the function "setBest" which takes as a parameter a pointer to type Person. Function "print" which prints the name of the person followed by the name of his best friend followed by the value of his popularity.

Write a main program that declares an array "list" of size 10. Each element of the array is a pointer to type Person. Open the file "persons.txt" that contains names of persons each on a separate line. Read the names of the persons and for each name create a new object and set a pointer of the array "list" to point to this person object. There is a second file "friends.txt" where every line contains the name of a person followed by the name of his best friend. Read each line and adjust the objects in "list" accordingly. In other words you have to set the best friend for an object and you have to increment the popularity of the corresponding person.

If "persons.txt" contains:

John

Steve

Mark

Mike

Ziad

Tarek

And if "friends.txt" contains:

John Steve
Steve Mark
Mark Steve
Mike John
Ziad John
Tarek Ziad

Then the program should output to the file "result.txt" the following information:

John	Best friend: Steve	Popularity = 2
Steve	Best friend: Mark	Popularity = 2
Mark	Best friend: Steve	Popularity = 1
Mike	Best friend: John	Popularity = 0
Ziad	Best friend: John	Popularity = 1
Tarek	Best friend: Ziad	Popularity = 0

GOOD LUCK!