UM-SJTU Joint Institute Final Examination
(vg101. Introduction to Computer and Programming, Fall 2016)
2-3:40pm, Dec. 13, 2016

General instructions

You exam will be solely on computer. You will be given a problem set (please download it from Canvas by yourself at the beginning of the exam) and you will need to design the program, implement it, compile it, debug it and test it on computer. Before the time is up, you should upload your solution in Canvas. You need to name the file using the format bellow: "(Your last name)(Initial of your first name)_sYourStudentID_final.zip" which includes all your source codes, For example, WuJ_s12345678_final.zip.

Unless otherwise indicated as part of the instructions for a specific problem, comments will not be required on the exam. Uncommented code that gets the job done will be sufficient for full credit on the problem. On the other hand, **comments may help you to get partial credit if they help us determine what you were trying to do**.

The examination is open-book, and you may make use of any texts, handouts, course notes, or any internet resources. However, you are not allowed to collaborate during the exam. Please close all communication software in your computer such as, QQ or any other chatting tools including email. **Any communication about the exam questions between you and other people is strictly prohibited**.

All of the following programs should be written in C or C++ and can be compiled and run using Visual Studio, or gcc. If you are using compilers other than Visual Studio, please indicate it in your codes as comments.

1. (10 points) Write a program to accept user input of a string, which composed of numbers and letters, then encrypt the string as follows: each character should be changed to its previous character, that is, 'a' should be changed to 'z', 'b' should be changed to 'a', '0' should be changed to '9', and '1' should be changed to '0', etc. For example, "Vg101zebra" should be encrypted to be "Uf090ydaqz". Print out your result after the encryption. You don't need to worry about invalid user inputs. The following is an example run of the program:

Please enter a string: Vg101zebra
The converted string: Uf090ydaqz

2. (20 points) Given an array of data, find the smallest data member (we will call it **percentile60**) that are greater than at least 60% of the data members. For example, the **percentile60** of {0 1 2 3 4 5 6 7 8 9} will be 6. Here we assume that any two data members in the array will not be the same, and there are at least 3 numbers in the array.

(1) (10) Write a function to return the **percentile60** of a *double* array with the following function head: *double GetPercentile60(double *parr, int n)*, where *parr* is a pointer pointing to the array, and *n* is the number of elements in the array.

(2) (10) Write the main function to accept user input of an array of data, and call your *GetPercentile60* function to get the **percentile60** and printout your result. Note that you don't know the number of elements in the array in advance and the user won't explicitly input the number. You don't need to worry about invalid user inputs. The following is an example run of the program:

Please enter the array: 1.1 5.2 7.4 4.2 3
The percentile60 is: 5.2

3. (20 points) Write a function with prototype "string AddNum(const string &num1, const string &num2, int b)" to add two numbers of base *b*, where the numbers can be very large and thus represented here as string objects. For example, AddNum("123456789123456789123456789", "123456789123456789123456789", 10) will add two decimal numbers and return "246913578246913578246913578", and AddNum("12A", "20", 16) will add two hexadecimal numbers and return "14A". Here we assume that $2 \leq b \leq 16$, and when $10 \leq b \leq 16$, we use 'A' - 'F' or 'a' - 'f' to represent 10-15. You should use the following main function to test your function:

```
int main()
{
        int base;
        string num1, num2;
        cout << "Please input base (2-16): ";
        cin >> base;
        cout << "Please input num1: ";
        cin >> num1;
        cout << "Please input num2: ";
        cin >> num2;
        cout << "num1 + num2 = " << AddNum(num1, num2, base) << endl;
        return 0;
}
```

4. (15 points) Use C++ to design and implement a class named MySpacetime to represent the spacetime event (x, y, z, ct). The class should contain the following member variables and functions (the members should be public unless explicitly specified):

- Four private data fields *x*, *y*, *z*, and *ct* (double type) that represent the space coordinates (*x*, *y*, *z*) and the time coordinate *ct*
- A constructor that constructs a spacetime event with four specified coordinates, all the coordinates have a default value of 0 if not specified. That is, we should be able to define an object of the class as "MySpacetime st1, st2(1, 2, 3, 4);" where st1.x, st1.y, st1.z, and st1.ct will all be 0, and st2.x is 1, st2.y is 2, st2.z is 3, and st2.ct is 4;
- A function named *setCoordinates* that sets data fields *x*, *y*, *z*, and *ct* according to user parameters. For example, suppose st is a MySpacetime object, then

"st.setCoordinates(1, 2, 3, 4)" will set st.x as 1, st.y as 2, st.z as 3, and st.ct as 4.

- A function named *getCoordinates* to get data fields *x*, *y*, *z*, and *ct*. For example, suppose st is a MySpacetime object, then "st.getCoordinates(x, y, z, ct)" will change the value of x, y, z, ct as st.x, st.y, st.z, and st.ct, respectively.
- A function named *distance* that returns the interval from this spacetime event to another spacetime event, here the interval is defined as

$$(x2-x1)^2+(y2-y1)^2+(z2-z1)^2-(ct2-ct1)^2$$

where (x1, y1, z1, ct1) and (x2, y2, z2, ct2) are the spacetime coordinates of the two events. For example, suppose st1, st2 are MySpacetime objects, then st1.distance(st2) will return the distance between st1 and st2.

Write the main function that read in user inputs of the (x, y, z, ct) coordinates of two events and display the interval between them. The following is an example run of the program:

Please input the coordinates of spacetime events:
First event: 0 0 0 0
Second point: 1.5 2.5 5 6
The interval is: -2.5

Additional instruction: don't worry about the precision of the output, you can just use the default output format for double type. And please write the class definition, implementation, and the main function in a single c++ file.

5. (35 points) (a) (5) Write a function with prototype "**string uint8ToBin(int x)**" to convert the integer x, where we assume 0 <= x <= 255, to a binary data represented as a **8-character** string object. For example, "uint8ToBin(127)" will return "01111111".

(b) (10) Write a function with prototype "**string FileToString(string sFileName)**" to read the binary file with file name **sFileName** and convert each byte of data **except the last two bytes** to a binary string using the function uint8ToBin, and then concatenate all the binary strings to get *str1*. Then convert the second-to-last byte in the file to an 8-character binary string and use the last *n* characters as *str2*, where *n* is indicated in the last byte of data in the file (we have 1 <= *n* <= 8). Finally return the concatenation of *str1* and *str2*. For example, assume the file contains five bytes of data "**247 122 208 10 7**" (in binary format in the file), then *str1* will be "**111101110111101011010000**" and *str2* will be "**0001010**", and thus the function will return the concatenation of *str1* and *str2* as "**11110111011110101101000000001010"**. You can use the provided file "P5-Encodedfile.dat" for testing.

(c) (20) The file "P5-HuffmanCode.txt" contains the codes for the space character and the 26 lower-case letters. For example, the line " :110" shows that the code for space character is 110, and the line "a:1010" shows that the code for letter a is 1010. Write a function with prototype "**string Decode(const string& sCode)**" and use the provided

code file to decode the string that you get from (b). For example, "Decode("**1111011101111010110100000001010")** will return "kia ora".

You should use the following main function for testing:

```
int main()
{
	// Test uint8ToBin
	int x;
	cout << "Please input a number among [0 255]: ";
	cin >> x;
	cout << uint8ToBin(x) << endl;
	// Test FileToString
	string sCode = FileToString("P5-Encodedfile.dat");
	cout << sCode << endl;
	// Test Decode
	cout << Decode(sCode) << endl;
}
```