UCLA Computer Science Department TA: Kung-Hua Chang

Student Name and ID ____________________________________

CS 31, SPRING 2014, PRACTICE MIDTERM II.

Problem	Maximal Possible Points	Received
1	10	
2	10	
3	10	
4	10	
5	10	
6	15	
Total Score	65	

Problem #1: The Collatz conjecture states that when taking any natural number n, if n is even, set n = n /2. If n is odd, set n = 3n + 1. Repeat this process until n is equal to 1. The conjecture states that no matter what number you start with, you will always reach 1. Please fill in the blanks below to complete the program. Assume n is from 0~100.

```
#include <iostream>
using namespace std;

int main()
{
        int n,count;
        while(cin >> n && n) {
                count = 0;
                while( n != 1 ) {

                        if(n%2 == 0) n /= 2;
                        else n = 3*n + 1;

                        count++;
                }
                cout << "There are " << count;
                cout << " numbers being generated!" << endl;
        }
}
```

Problem #2: An National Spying Agency worker has found some patterns in encoded messages such that when 2 consecutive encoded messages (shorter than 9 characters) with complementing 0s and 1s show up, the real message, though encoded, follows. He needs your help to write the following program for him to check for such patterns. For example,
"1111" and "0000" are complementary to each other, while
"1011" and "0101" are not complementary to each other.

```
#include <iostream>
#include <cstring>
using namespace std;

bool isComplement(char row1[],int len1,char row2[],int len2)
{

        if(len1 != len2)
                return false;

        for(int i=0 ; i < len1; i++)
                if( row1[i] == row2[i] )
                        return false;
        return true;

}

int main()
{
        char row1[9],row2[9];
        cin >> row1 >> row2;
        if(isComplement(row1,strlen(row1),row2,strlen(row2)))
                cout << "They are complementary to each other!" << endl;
        else
                cout << "Life is tough!!" << endl;
}
```

Problem #3: An old trick to encrypt messages is to embed the real message into random characters in a certain pattern. Suppose that we know the pattern to decode is to extract the 1st, 3rd, 5th, 7th, …characters out of the encoded messages. For example, “BAAANPALNEA” can be decoded as “BANANA”. Please complete such function below. Assume that encodedMsg will always have length greater than 1.

```
#include <iostream>
#include <cstring>
using namespace std;

void solvePuzzle( char encodedMsg[] ,char decodedMsg[] )
{
        int i, j;

        for(i=0, j=0 ; i < strlen(encodedMsg) ; i+=2, j++)
                decodedMsg[j] = encodedMsg[i];

        decodedMsg[j] = 0;

}

int main()
{
        char encodedMsg[] = "BAAANPALNEA";
        char decodedMsg[100];

        solvePuzzle(encodedMsg,decodedMsg);

        cout << "The decoded message is: " << decodedMsg << endl;

        return 0;
}
```

Problem #4: What is the output of the program below? (1)

```
#include <iostream>
#include <cstring>
using namespace std;

bool findTriple(char str[])
{
        int i;

        bool foundTripleZero = false;
        for(i = 1 ; i < strlen(str)-2 ; i++ )
        {
                if(foundTripleZero == true)
                        foundTripleZero = false;

                else if( ( str[i] == str[i+1]) && (str[i+1] == str[i+2]) )
                        foundTripleZero = true;
        }
        return foundTripleZero;
}

int main()
{
        cout << findTriple("01110");
        cout << findTriple("00110");
        cout << findTriple("00010");
        cout << findTriple("10001");
        cout << endl;
}
```

(1) 0000
(2) 0001
(3) 0010
(4) 0011
(5) 1000
(6) 1001
(7) 1010
(8) 1011
(9) 1100

Problem #5: You work in a math lab that requires finding root x for a function f(x) such that f(x) = 0. Please complete the following findRoot function without using any extra variables. You need to check if the values from x array can make f(x) equal to zero and if so, store such value in the root array and update m to signal how many values are in the root array. findRoot() should return true if there is at least a value in x array such that f(x) is equal to 0.

```
#include <iostream>
using namespace std;
int f(int x)
{
        // x^3 - 2x^2 - 5x + 6
        return (x*x*x-2*x*x-5*x+6);
}
bool findRoot(int x[], int n, int root[], int &m)
{
        bool foundRoot = false;
        int i;
        m = 0;
        for(i=0;i<n;i++)
                if( f( x[i] ) == 0) {
                        root[m] = x[i];
                        m++;
                        foundRoot = true;
                }
        return foundRoot;
}
int main()
{
        int x[31],root[31];
        int n = 31,m = 0;

        for(int i=0;i < n ; i++)
                x[i] = i - 15;

        if(findRoot(x,n,root,m))
                cout << "Found root for f(x)" << endl;
}
```

Problem #6: You are to write a program to control a robot to move left and right. The up and down keys, though can be accepted by the program, are no-op, but they are still valid keys in the commands. Your goal is to complete the function executeCommands such that the program

(1) Checks if the keys are valid (L / R / U / D) and if any of keys in the string cmdStr are not valid, return -1 without further processing.
(2) Updates points if the robot lands on a specific position as determined by updatePoints() function. The starting point (the initial value in pos in executeCommands function) can get points right away. Even if the robot does not move because the command is U or D, the robot can still get points by staying at that position.
(3) Update position of the robot by adding 1 to pos if the key is R, subtracting 1 to pos if the key is L, and do nothing if the key is U or D.
(4) After you process a valid key, update points.
(5) Return the points back from executeCommands function.

You cannot use any extra variables or arrays in executeCommands function.

To verify the correctness of your program, the output in the completed program should be

```
0
-1
-1
9
```

Assume the program below already has the necessary headers:

```
#include <iostream>
#include <string>
using namespace std;
```

```
int updatePoints(int pos)
{
        if( pos == 3 ) return 1;
        if( pos == 1 ) return -1;
        if( pos == 5 ) return 2;
        if( pos == 2 ) return -2;
        return 0;
}
int executeCommands(string cmdStr, int pos, int points)
{
        int i;

        for(i = 0 ; i < cmdStr.size() ; i++ )
                if( cmdStr[i] !='U' && cmdStr[i] !='D' &&
                   cmdStr[i] != 'L' && cmdStr[i] != 'R' )
                        return -1;

        points = points + updatePoints(pos);

        for(i = 0 ; i < cmdStr.size() ; i++ )
        {
                if( cmdStr[i] == 'L' ) pos--;
                else if( cmdStr[i] == 'R' ) pos++;

                points = points + updatePoints(pos);
        }
        return points;
}
int main()
{
        string command1 = "LULDR";
        string command2 = "LRDULRDUXX";
        string command3 = "UUDDLLRRBA";
        string command4 = "DDURDDUR";

        cout << executeCommands(command1, 0 , 0) << endl;
        cout << executeCommands(command2, 1 , 1) << endl;
        cout << executeCommands(command3, 2 , 2) << endl;
        cout << executeCommands(command4, 3 , 3) << endl;
}
```