

NAME: FIRST LAST

SEAT ROW

NUMBER

1. (30 points) Answer the following questions.

- (1) Declare an array of doubles called grades with size 3. Initialize the elements, from the first one to the last one, to be 1.2, 3.6, 1.1.

```
double grades[] = {1.2, 3.6, 1.1};
```

Or

```
double grades[3] = {1.2, 3.6, 1.1};
```

Or

```
double grades[3];
```

```
grades[0] = 1.2;
```

```
grades[1] = 3.6;
```

```
grades[2] = 1.1;
```

- (2) Declare function foo whose input parameter is an int (pass by value) and return is a char. You just need to write the function header, no implementation is needed.

```
char foo(int n);
```

- (3) Write code to generate a random floating point number in [-2, 2].

- (1) To generate a random floating number in [0, 1], use $1.0 * \text{rand}() / \text{RAND_MAX}$. Note that $1.0 * \text{rand}()$ converts $\text{rand}()$ to double type, otherwise $\text{rand}() / \text{RAND_MAX}$ returns either integer 0 or integer 1.

- (2) To generate a random floating number in [0, 4], since the distance between -2 and 2 is 4, is $1.0 * \text{rand}() / \text{RAND_MAX} * 4$, or simplified as $4.0 * \text{rand}() / \text{RAND_MAX}$.

- (3) To generate a random floating number in [-2, 2] is to shift the result in (2) by -2, that is, $-2 + 4.0 * \text{rand}() / \text{RAND_MAX}$.

Complete code is `double num = -2 + 4.0 * rand() / RAND_MAX;`

- (4) Given array of strings as follows

```
string greetings[] = {"Hi", "Hello", "Wow"};
```

What is the value for `greetings[2][1]`?

`greetings[2]` is string "Wow", `greetings[2][1]` is the letter indexed at 1, which is the second letter (the first letter is indexed at 0), so the answer is letter 'o'.

(5) What is the default compiled runnable file generated if a C++ source code is successfully compiled?

a.out

(6) What is the output of the following code?

```
int value = 0;
for (int i = 1; i <= 6; i += 5)
{
    value += i;
    cout << value << endl;
}
```

In the very beginning value is 0.

variable i	condition i <= 6	value += i	cout << value << endl	i += 5
1	Yes	value becomes 1	print 1 in a line	i becomes 6
6	Yes	value becomes 7	Print 7 in a line	i becomes 11
11	No			

Output

1
7

(7) Write code to calculate a^{-5} , where a is an int that is not zero.

```
double value = pow(a, -5);
```

Or

```
double value = 1;
```

// a^{-5} is the same as $(1.0/a)^5$, since a is an int, we use 1.0 / a get decimal numbers

```
for (int i = 0; i < 5; i++)
```

```
    value *= 1.0 / a;
```

(8) If m is 5 and n is 3, what is the value of $1 / 2 * m * n$?

Operators / and * have the same precedence, and they run from left to right. $1 / 2$ is integer division, so the result is 0. Value of $1 / 2 * m * n$ is 0.

(9) Suppose n is an int, write code to extract its hundreds digit. For example, suppose n is 21, after your code, you get 0; if n is 1265, then your code gets 2.

```
int hundreds = n / 100 % 10;
```

n / 100 returns n without the last two digits, for example, when n is 1265, then n / 100 returns 12, to get the hundreds digit, that is 2 in this example, we need to extract the last digit from n / 100, so we need to run n / 100 % 10.

(10) What is the output of the following code?



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







void foo(int size);

int main()
{
    foo(3);
    return 0;
}

void foo(int size)
{
    for (int numAsts = size; numAsts >= 1; numAsts--)
    {
        for (int i = 0; i < size - numAsts; i++)
            cout << " ";
        for (int i = 0; i < numAsts; i++)
            cout << "*";
        cout << endl;
    }
}
```

Here is an analysis when size is 3.

numAsts	numAsts ≥ 1	for (int i = 0; i < size - numAsts; i++) cout << " "; print " " (size-numAsts) times	for (int i = 0; i < numAsts; i++) cout << "*"; print "*" numAsts times	cout << endl;	numAsts- -
3	yes	Print zero space character. That is, do not print any space.	Print three *'s, that is, ***	Print a new line, the screen looks like ***	2
2	yes	Print one space character. That is, 	Print two *'s, that is, **	Print a new line, the screen look like ***  **	1

				Where  Denotes a space.	
1	yes	Print two space characters, that is,  	Print one *'s, that is, *	Print a new line, the screen look like ***  **   *	0
0	no				

The output is

```
***
**
*
```

2. (45 points) Short programming exercises

(2.1) Write code in main that generate 10 random integers in [20, 100], tally how many of grades are ≥ 80 , and how many grades are between 60 (included) and 80 (not included), and how many grades are < 60 .

```
#include <iostream>
#include <cstdlib> //rand,
#include <ctime> //use time
using namespace std;

//Write code in main that generate 10 random integers
//in [20, 100], tally how many of grades are  $\geq 80$ ,
//and how many grades are between 60 (included) and 80
//(not included), and how many grades are  $< 60$ .

//keys:
//(1) generate 10 random integers
//for (int i = 0; i < 10; i++)
//    generate a random int in [20, 100]
//(2) generate a random int in [20, 100] is
//    score = rand() % (100 - 20 + 1) + 20;
//    assume that score is declared as an int.
//    There are a total of 100 - 20 + 1 = 81 integers
//    in [20, 100],
//    expression rand() % (100 - 20 + 1) returns
//    a random int in [0, 80].
//    expression rand() % (100 - 20 + 1) + 20 returns
//    a random int in [20, 100].
//(3) Before the loop, set number of each category
//    to be 0.
//    Use nested if-else statement to
//    tally each category.

int main()
{
    srand(time(NULL));
    //Use the current time as a seed
    //to generate random int.

    int score;
    int numGE80 = 0; //score  $\geq 80$ 
    int num60To80 = 0; //score  $\geq 60$  && score  $< 80$ 
    int numLT60 = 0; //score  $< 60$ 
    for (int i = 0; i < 10; i++)
    {
        //Generate a random int in [20, 100].
        score = rand() % (100 - 20 + 1) + 20;
```

```
    if (score >= 80)
        numGE80++;
    else if (score >= 60)
        num60To80++;
    else numLT60++;
}

//print out tallied result (optional)
cout << "score >= 80: " << numGE80 << endl;
cout << "score >= 60 and score < 80: "
    << num60To80 << endl;
cout << "score < 60: " << numLT60 << endl;
return 0;
}
```

(2.2) Define a function, sort two given strings according to their dictionary order, return type is void. For example, suppose string a is "seller" and string b is "buyer", after calling this function, a is changed to "buyer" and b is changed to "seller". Suppose a is "success" and b is "work", then after the function, a and b do not change.

```
#include <iostream>
#include <cassert> //assert
using namespace std;

//Define a function, sort two given strings
//according to their dictionary order,
//return type is void.
//For example, suppose string a is "seller" and
//string b is "buyer", after calling this function,
//a is changed to "buyer" and b is changed to "seller".
//Suppose a is "success" and b is "work",
//then after the function, a and b do not change.

void sort(string& a, string& b);

int main()
{
    string a("seller");
    //initialize a string a by "seller",
    //you can write as
    //string a = "seller"; as well.
    //string is few (if not only) type in C++
    //that can be initialized as a class constructor
    //like type objectName(parameter_in_constructor)
    //or like a primitive type use
    //type variableName = value_of_that_type;

    string b("buyer");

    cout << "Before sorting" << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    sort(a, b);

    cout << "After sorting" << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    a = "success";
    b = "work";
}
```



```

cout << "\nBefore sortping" << endl;
    //\n means new line
cout << "a = " << a << endl;
cout << "b = " << b << endl;

sort(a, b);

cout << "After sorting" << endl;
cout << "a = " << a << endl;
cout << "b = " << b << endl;

//Let assert function do an auto check.
assert(a == "success" && b == "work"
    && a <= b);

return 0;
}

void sort(string& a, string& b)
{
    if (a > b) //a > b means a is after b in dictionary
    {
        string temp = a;
        a = b;
        b = temp;
    }
}

```

(2.3) Write code.

Create an ifstream object fin that reads text file named data.txt.

```
ifstream fin("data.txt");
```

Write code to ignore the first line of data.txt.

Hints: you might want to use ignore method of file stream.

```
istream& ignore (streamsize n = 1, int delim = EOF);
```

Extract and discard characters

Extracts characters from the input sequence and discards them, until either n characters have been extracted, or one compares equal to *delim*.

```
fin.ignore(INT_MAX, '\n');
```

Suppose data.txt has only a column of decimal numbers. Write code to find out and print the maximum value.

```
double value;
fin >> value; //read the first data

if (fin.fail()) //no data
    return 0;

//Initialize max with one of its own data
//Warning: do not set double max to be an arbitrary
//value, for example, 0.
//Then if all data are negative,
//their maximum cannot be 0.
double max = value;
    //use the first data to initialize max
while (fin >> value)
{
    if (value > max)
        max = value;
}

cout << "max = " << max << endl;
```

Call close method of fin.

```
fin.close();
```

A complete code is as follows.

```
#include <iostream>
#include <fstream> //ifstream, ofstream
using namespace std;

//Here is an example of contents of data.txt
//scores
//92.6
//25.1
//53.6

//Sample output:
//max = 92.6
int main()
{
    ifstream fin("data.txt");
        //fin opens file "data.txt" to read.

    //skip the first line, which is just column names.
    //Skip INT_MAX letters or till meeting '\n',
    //whichever is met first.
    fin.ignore(INT_MAX, '\n');

    double value;
    fin >> value; //read the first data

    if (fin.fail()) //no data
        return 0;

    //Initialize max with one of its own data
    //Warning: do not set double max to be an arbitrary
    //value, for example, 0.
    //Then if all data are negative,
    //their maximum cannot be 0.
    double max = value;
        //use the first data to initialize max
    while (fin >> value)
    {
        if (value > max)
            max = value;
    }

    cout << "max = " << max << endl;

    fin.close(); //finish reading the file opened by fin
    return 0;
}
```

3. (10 points) **Define a function**, for a given string `str`, return true if all its characters are either '0' or '1', otherwise, return false.

```
#include <iostream>
#include <cassert> //assert
using namespace std;

//Define a function, for a given string str,
//return true if all its characters are only
//letters '0', '1', otherwise, return false.

bool only01(string str);

int main() //optional
{
    assert(only01("01010") == true);
        //only01("01010") is expected to return true.
    assert(only01("01202") == false);
        //only01("01202") is expected to return false.
    assert(only01("") == false);
    assert(only01("0123") == false);
    return 0;
}

bool only01(string str)
{
    int len = str.length();

    if (len == 0) //handle special case when str is "".
        return false;

    for (int i = 0; i < len; i++)
        //return false for first letter that is
        //neither '0' nor '1'.
        if (str[i] != '0' && str[i] != '1')
            return false;

    //Test every letter in str,
    //and it is one of '0' or '1',
    //otherwise, we would have returned false
    //in one of the above for-loop.
    return true;
}
```

4. (15 points) Define a function, for a given int, return how many digits are 1. For example, if the given integer is 20, then the number of 1's is 0, if the given integer is 101, then the number of 1's is 2.

```
#include <iostream>
#include <cassert> //assert
using namespace std;

//Define a function, for a given int,
//return how many digits are 1.
//For example, if the given integer is 20,
//then the number of 1's is 0,
//if the given integer is 101,
//then the number of 1's is 2.

int num1s(int n);

int main()
{
    assert(num1s(20) == 0);
    assert(num1s(-21) == 1);
        //expected return of num1s(-21) is 1.
    assert(num1s(-300) == 0);
        //expected return of num1s(-300) is 0.
    assert(num1s(-123) == 1);
    assert(num1s(101) == 2);

    return 0;
}

//key: throw the last digit away a time,
//until no more digit to throw away.
//Before throw away the last digit,
//check whether it is 1 or not.
//Each integer has at least a digit to be throw away,
//so we use do-while statement.
//Note that -231 % 10 returns -1,
//so need to consider when n is negative.
int num1s(int n)
{
    if (n < 0) //-132 % 10 returns -2,
        //we would not like to handle negative number
        //in the following code.
        n *= -1;

    int num1s = 0;
    int lastDigit;
```

```
do {  
    lastDigit = n % 10;  
    //save last digit of n to lastDigit  
  
    if (lastDigit == 1)  
        num1s++;  
  
    n /= 10; //throw away the last digit from n  
  
} while (n != 0);  
  
return num1s;  
}
```