



2023 年度全国中学生程序设计竞赛 - 第五次工作坊

National Secondary School Programming Contest - Workshop 5

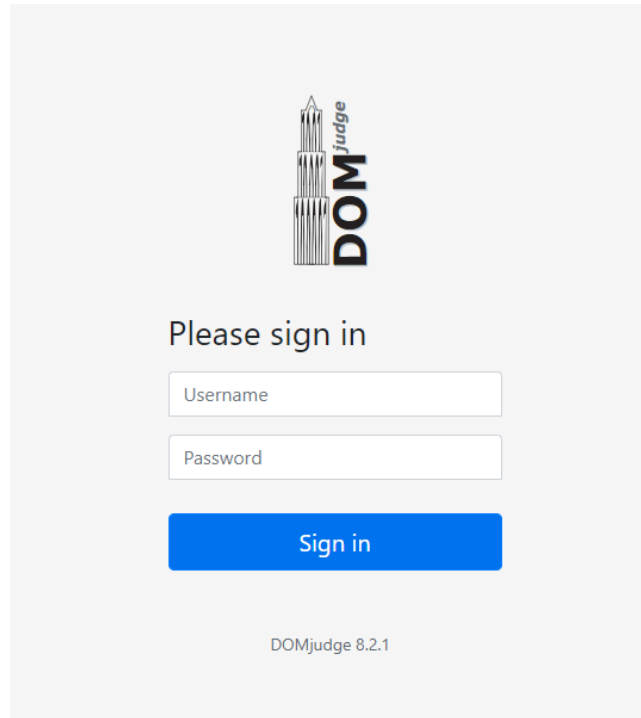
Speaker: **Dr. Jasmina Khaw Yen Min**

Date: **6 AUGUST 2023**

Overview of Workshop

Session	Time	Details
Morning (First Session)	9:00AM – 10:30AM	- Problem Solving 1
Morning (Second Session)	10:30PM – 12:00PM	- Problem Solving 2

Getting into the competition (Demo)



The screenshot shows the login interface for DOMjudge. At the top left is the DOMjudge logo, which consists of a stylized building icon and the text 'DOMjudge'. Below the logo, the text 'Please sign in' is displayed. There are two input fields: 'Username' and 'Password'. Below these fields is a blue 'Sign in' button. At the bottom of the page, the version number 'DOMjudge 8.2.1' is visible.

<http://nsspc2023.utar.edu.my/login>

- username: demo01, password: 1234567890
- username: demo02, password: 1234567890
- username: Demo03, password: 1234567890
- username: demo04, password: 1234567890

Type and Area of Triangle (15 marks):

Find the area of a triangle with the lengths of three sides, and determine which of the following types that the triangle belongs to:

- a. Scalene Triangle
- b. Isosceles Triangle
- c. Equilateral Triangle

Write a program to

Input, 3 floating-point values, side1, side2, side3.

Output, in sequence,

the type of the triangle (Scalene Triangle, Isosceles Triangle, or Equilateral Triangle) in the first line, and the area of the triangle, rounded to an integer, in the second line.

Note: Display “Invalid” if the triangle cannot be formed from the three inputs of lengths.

Examples (例子)

Input (输入)	Output (输出)
3 4 5	Scalene Triangle 6
36.6 17.0 40	Scalene Triangle 311
8 8 16	Invalid
6 6.0 11	Isosceles Triangle 13
67.00 67 67.0	Equilateral Triangle 1944

Suggested Steps:

Method: Using Side Lengths

1. Calculate the semiperimeter of the triangle

$$\text{semiperimeter, } s = \frac{1}{2}(a+b+c)$$

2. Set up Heron's formula

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

For example:

If $a=3$, $b=4$ and $c=5$:

$$s = \frac{1}{2}(3 + 4 + 5)$$
$$s = \frac{1}{2}(12) = 6$$

$$\text{Area} = \sqrt{6(6-3)(6-4)(6-5)}$$

$$\text{Area} = \sqrt{6(3)(2)(1)}$$

$$\text{Area} = \sqrt{6(6)}$$

$$\text{Area} = \sqrt{36}$$

$$\text{Area} = 6$$

Suggested Solution:

First Part:

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

int main(){
    float side1, side2, side3;

    cin >> side1 >> side2 >> side3;

    float s = (side1+side2+side3)/2;
    float area = sqrt(s*(s-side1)*
        (s-side2)*(s-side3));

    if(side1>0 && side2 >0 && side3>0 &&
        side1+side2>side3 &&
        side1+side3>side2 &&
        side2+side3>side1){
```

Second Part:

```
    if(side1 == side2 && side2 == side3)
        cout << "Equilateral Triangle";

    else if(side1 == side2 || side2 ==
        side3 || side1 == side3)
        cout << "Isosceles Triangle";
    else
        cout << "Scalene Triangle";
    cout<<endl<<fixed<<setprecision(0)<<area;
}
else
    cout<<"Invalid"<<endl;

return 0;
}
```

Pairing Integers (20 marks):

A secondary school teacher, Mr. Jimmy, gives an integer, X , between -200 and 200 (inclusive) to his students in the class. Then, he picks a student to randomly provide a sequence of integers with the requirement that each integer must be in the range from -100 to 100 (inclusive). If any integer provided by the student is not in the range, then Mr. Jimmy will just skip it and ask the student to provide more integer(s) until he collects twenty integers that are in the required range, i.e., $-100 \leq a_n \leq 100$ for $1 \leq n \leq 20$.

Mr. Jimmy then lists out all these twenty valid integers on a whiteboard. The students in the class are now required to find out the total number of distinct pairs of valid integers which have the sum equal to X .

For example:

If a given integer is -101 , it should be skipped because it does not meet the requirement set by Mr. Jimmy.

On the other hand, if $X=50$, then $(1, 49)$ and $(49, 1)$ are considered the same pair, and $(1, 49)$ and $(53, -3)$ are considered the distinct pairs of integers that fulfil the sum equal to 50 .

Pairing Integers (20 marks): (cont'd)

Write a program to

Input, in sequence, a series of integers with the following requirements :

The first integer is X , where $-200 \leq X \leq 200$;

subsequently, a series of integers where your programme needs to collect the first twenty of them that meets the requirement $-100 \leq a_n \leq 100$, where $1 \leq n \leq 20$.

Output the total number of distinct pairs of integers (a_i , a_j) , which fulfils $1 \leq i, j \leq 20$, $i \neq j$, and $a_i + a_j = X$.

Examples (例子)

Input (输入)	Output (输出)		Input (输入)	Output (输出)
55	6		80	5
2			50	
54			66	
-70			20	
30			33	
15			99	
40			100	
20			-20	
10			7	
35			27	
23			-5	
22			73	
60			40	
-15			50	
8			40	
-5			-20	
45			14	
5			100	
33			30	
1			44	
0			40	

Examples (例子) (cont'd)

Input (输入)	Output (输出)		Input (输入)	Output (输出)
73	4		-78	5
2			12	
56			70	
73			27	
88			0	
34			55	
5			-100	
-6			-90	
50			78	
17			-39	
0			60	
15			18	
102			-56	
-29			-22	
40			-17	
68			30	
10			-61	
230			88	
-20			-10	
93			-68	
7			-39	
3				
44				

Highlighted Possible Pairs

Input (输入)	Output (输出)		Input (输入)	Output (输出)
55	6		80	5
2			50	
54			66	
-70			20	
30			33	
15			99	
40			100	
20			-20	
10			7	
35			27	
23			-5	
22			73	
60			40	
-15			50	
8			40	
-5			-20	
45			14	
5			100	
33			30	
1			44	
0	40			

Highlighted Possible Pairs (cont'd)

Input (输入)	Output (输出)		Input (输入)	Output (输出)
73	4		-78	5
2			12	
56			70	
73			27	
88			0	
34			55	
5			-100	
-6			-90	
50			78	
17			-39	
0			60	
15			18	
102			-56	
-29			-22	
40			-17	
68			30	
10			-61	
230			88	
-20			-10	
93			-68	
7			-39	
3				
44				

Suggested Solution:

First Part:

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

int main(){
    int array1[20], y=0;
    int i=0, sum, n=20, j=0, k, ctr=0;
    do
        cin>>sum;
    while(sum < -200 || sum > 200);

    while(i<20){
        cin>>array1[i];
        if (array1[i]>=-100 && array1[i]<=100)
            i++;
    }
```

Second Part:

```
//check duplicate and remove
for (i = 0; i < n; i++){
    for (j = i + 1; j < n;){
        if (array1[j] == array1[i] &&
            array1[j]+array1[i] != sum){
            for (k = j; k < n; k++){
                array1[k] = array1[k + 1];
            }
            n--;
        }
        else
            j++;
    }
}
```

Suggested Solution:

Third Part:

```
//list of pairs
for (int m=0; m<n; m++){
    for (int j=m+1; j<n; j++){
        if (array1[m]+array1[j] == sum){
            //skip for displaying two same values pair more than once
            if(array1[m]==array1[j])
                y++;
            if(y>1)
                continue;
            else
                ctr++;
        }
    }
}
cout<<ctr<<endl;
return 0;
}
```

Finding Vocabularies (30 marks):

During an English writing lesson, students are given a paragraph selected from a story book. Suppose that the number of words in the paragraph is not more than 100. Students need to find out all the vocabularies that contain at least three vowels, and then convert these vocabularies in uppercase and show them on the whiteboard. The students are also required to count the number of vocabularies found in this category. Note that combining two words with a hyphen (-) or apostrophe (') is considered as one word. Moreover, repeated vocabulary must be ignored and not counted.

(Hint: The word "competition" contains 5 vowels. Meanwhile, "bed-time" and "they've" are respectively considered as one word.)

Write a program to

Input, a paragraph selected from a story book which consists of n words, where $1 \leq n \leq 100$.

Output, in sequence

- (1) the list of vocabularies from the paragraph that contain at least three vowels (vowels can be duplicated); note that these vocabularies must be shown **in uppercase** and listed in the sequence as how they appear in the selected paragraph.
- (2) the number of vocabularies found in this category.

Note:

1. Hyphen (-) and apostrophe (') are considered as part of the word, and thus they must not be removed.
2. Repeated vocabulary **must** be ignored and not counted.

Example (例子)

Input (输入)	Output (输出)
I am sorry to say that Peter was not very well during the evening. His mother put him to bed, and made some camomile tea; and she gave a dose of it to Peter! "One table-spoonful to be taken at bed-time."	EVENING CAMOMILE TABLE-SPOONFUL BED-TIME 4
"Someone's been sitting in my chair and they've broken it to pieces," cried the Baby bear. They decided to look around some more and when they got upstairs to the bedroom, "Someone's been sleeping in my bed and she's still there!" exclaimed the Baby bear.	SOMEONE'S PIECES DECIDED AROUND UPSTAIRS BEDROOM SLEEPING EXCLAIMED 8
We approached the last hundred metres. I was only half a motorcycle's length behind him. With a final flick, we crossed the finish line. We looked up at the giant screen. A camera replay was being shown. It was too close to call a winner. "And the winner is Will!" the commentator boomed through the sound system.	APPROACHED MOTORCYCLE'S LOOKED CAMERA COMMENTATOR BOOMED 6

Example (例子) (cont'd)

Input (输入)	Output (输出)
They say that a fool and his money are soon parted. Thus, I would be very careful with how I spend my money. I would invest my money with my parents' advice and keep some for rainy days.	CAREFUL ADVICE 2
The other, astonished to be addressed so familiarly by this common woman, did not recognize her. She stammered: "But, madame, I don't know. You must have made a mistake."	ASTONISHED ADDRESSED FAMILIARLY RECOGNIZE STAMMERED MADAME MISTAKE 7

Suggested Steps:

Steps:

1. Read each word from the input paragraph.
2. Remove punctuation but not hyphen (-) or apostrophe (').
3. Check if the word has 3 or more vowels.
4. Convert each word to uppercase.
5. If word not seen before, add to list of words.
6. Print all words that has 3 or more vowels.
7. Print the number of words with 3 or more vowels found.

Suggested Solution:

First Part:

```
#include <iostream>
#include <sstream>
#include <unordered_map>
#include <vector>
#include <algorithm>
#include <string>
#include <cctype>
using namespace std;

// Function to check if a character is vowel
bool isVowel(char c) {
    c = tolower(c); // convert to lowercase
    return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
}
```

Second Part:

```
// Function to count the number of vowels in a word
int countVowels(const string& word) {
    int count = 0;
    for (char c : word) {
        if (isVowel(c)) {
            ++count; // increment count if character is vowel
        }
    }
    return count;
}
```

Suggested Solution: (cont'd)

Third Part:

```
// Function to remove punctuation from a word  
// Preserves hyphens and apostrophes  
string removePunctuation(string word) {  
    word.erase(remove_if(word.begin(), word.end(), [](char c) {  
        return ispunct(c) && c != '-' && c != '\''; // remove if punctuation but not hyphen or apostrophe  
    }), word.end());  
    return word;  
}  
  
int main() {  
    string paragraph;  
    getline(cin, paragraph); // read the paragraph  
    istringstream iss(paragraph);  
    unordered_map<string, bool> seen; // track seen words  
    vector<string> words; // store words in order  
    string word;
```

Suggested Solution: (cont'd)

Forth Part:

```
while (iss >> word) { // read each word
    string newWord = removePunctuation(word); // remove punctuation
    if (countVowels(newWord) >= 3) { // if word has 3 or more vowels
        transform(newWord.begin(), newWord.end(), newWord.begin(), ::toupper); // convert to uppercase
        if (!seen[newWord]) { // if word not seen before
            words.push_back(newWord); // add to list of words
            seen[newWord] = true; // mark word as seen
        }
    }
}

for (const string& word : words) { // print all words
    cout << word << '\n';
}
cout << words.size() << '\n'; // print count of words

return 0;
}
```

Cipher Algorithm (40 marks)

A **cipher algorithm** is a mathematical formula designed specifically to obscure the value and content of the data so that unintended recipients cannot understand it. In this question, you are required to write a program which is able to use a given keyword to encrypt an original string to an encrypted string, or, decrypt an encrypted string back to the original string by using the same keyword.

The encryption principle is explained as follows.

- 1) First of all, each alphabet is corresponding to a number by referring to the following table.

Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M
Number	0	1	2	3	4	5	6	7	8	9	10	11	12
Alphabet	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number	13	14	15	16	17	18	19	20	21	22	23	24	25

- 2) The system will then be given an original string and a keyword with different lengths.
- 3) To encrypt the original string, the keyword will be repeated indefinitely until the keyword string has the same length as that of the original string, so that there is a one-to-one correspondence between the original string and the keyword string

Cipher Algorithm (40 marks) (cont'd)

- 4) Then, at each position in the strings, the number of the original character is added to the number of the corresponding keyword character.
- 5) The sum from (4) will then be divided by 26 in order to get the modulo (i.e., remainder). With this modulo number, the encrypted character can then be obtained.

For example, if the string is “HOWAREYOUTODAY”, the keyword is “IMFINE”. The encrypted string will be “PABIEIGAZBBBHIK” based on the following table.

Original String	H	O	W	A	R	E	Y	O	U	T	O	D	A	Y
Keyword String	I	M	F	I	N	E	I	M	F	I	N	E	I	M
Encrypted String	P	A	B	I	E	I	G	A	Z	B	B	H	I	K

Please note that in the second row of the above table, the keyword is repeated until the keyword string has the same length as that of the original string.

Then each encrypted character is obtained by adding the number of the original character and the number of the corresponding keyword character. If the sum exceeds 26, then the sum will be subtracted by 26 in order to get the encrypted character.

For example, the 1st alphabet from the original string is “H”, which corresponds to number 7, and the 1st alphabet from the keyword string is “I”, which corresponds to number 8. The sum of 7 and 8 is 15, which results in the encrypted alphabet “P”.

Cipher Algorithm (40 marks) (cont'd)

For another example, with reference to the 10th column of the above table, alphabet “T” from the original string corresponds to number 19, and alphabet “I” from the keyword string corresponds to number 8. When 19 and 8 are added, the sum is 27. Since there is no number 27 in the table, it should be subtracted by 26, resulting in number 1, which is corresponding to the alphabet “B”.

Write a program to

Input three lines as described below.

The first line is the string to be encrypted or decrypted.

The second line is the keyword to be used for encryption and decryption.

The third line is a single alphabet, either “E” or “D”. If it is an “E”, the programme needs to encrypt the string given in the first line by using the keyword in the second line. However, if it is a “D”, the programme needs to decrypt the string in the first line by using the keyword in the second line.

Note: The inputs in the first and second lines are limited to the alphabets in uppercase, each line contains maximum up to 20 alphabets.

Output the encrypted/decrypted string for the string given in the first line, by using the keyword as instructed. If any of the input strings is more than 20 alphabets, or contains undefined character(s)/symbol(s), or the single alphabet in the third line is neither “E” nor “D”, then output “Invalid Input”.

Example (例子)

Input (输入)	Output (输出)
HELLOWORLD GOOD E	NSZOUKCURR
RSVYCAQEYMSVBEQ YES D	TODAYISAGOODDAY
LETSSTARTTOCODE OKA* E	Invalid Input
GOODBYESEYOU SURE K	Invalid Input
HSVVPHCIMSNX LOVE D	WEARETHEBEST

Suggested Steps:

Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M
Number	0	1	2	3	4	5	6	7	8	9	10	11	12
Alphabet	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number	13	14	15	16	17	18	19	20	21	22	23	24	25

Example 1 (Encryption):

Original String	H	E	L	L	O	W	O	R	L	D
Keyword String	G	O	O	D	G	O	O	D	G	O
Encrypted String	N	S	Z	O	U	K	C	U	R	R

- H=7, G=6 Hence, $7+6=13 \Rightarrow N$
- E=4, O=14 Hence, $4+14=18 \Rightarrow S$
- L=11, O=14 Hence, $11+14=25 \Rightarrow Z$
- L=11, D=3 Hence, $11+3=14 \Rightarrow O$
- O=14, G=6 Hence, $14+6=20 \Rightarrow U$
- W=22, O=14 Hence, $22+14=36 \Rightarrow 36\%26=10 \Rightarrow K$...

Example 2 (Decryption):

Original String	R	S	V	Y	C	A	Q	E	Y	M	S	V	B	E	Q
Keyword String	Y	E	S	Y	E	S	Y	E	S	Y	E	S	Y	E	S
Decrypted String	T	O	D	A	Y	I	S	A	G	O	O	D	D	A	Y

- R=17, Y=24 Hence, $17-24=-7 \Rightarrow -7+26=19 \Rightarrow T$
- S=18, E=4 Hence, $18-4=14 \Rightarrow O$
- V=21, S=18 Hence, $21-18=3 \Rightarrow D$
- Y=24, Y=24 Hence, $24-24=0 \Rightarrow A$
- C=2, E=4 Hence, $2-4=-2 \Rightarrow -2+26=24 \Rightarrow Y$...
- A=0, S=18 Hence, $0-18=-18 \Rightarrow -18+26=8 \Rightarrow I$

Suggested Steps: (cont'd)

Steps:

1. Read each character from the input.
2. Check each character in the input and keyword for validity
 - 1st line: string to be encrypted or decrypted (only uppercase and **limited to 20 characters**)
 - 2nd line: keyword to be used for encryption and decryption (**only uppercase and limited to 20 characters**)
 - 3rd line: single alphabet, either "E" or "D".
3. Perform the encryption or decryption and print the result

Suggested Solution:

First Part:

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

// Function to check if a character is not an uppercase
letter
bool isInvalidChar(char c) {
    return c < 'A' || c > 'Z';
}

// Function to add two characters, with wraparound
from 'Z' to 'A'
char addChar(char a, char b) {
    return (a - 'A' + b - 'A') % 26 + 'A';
}

// Function to subtract one character from another, with
wraparound from 'A' to 'Z'
char subtractChar(char a, char b) {
    return ((a - 'A') - (b - 'A') + 26) % 26 + 'A';
}
```

Second Part:

```
// Function to transform a string by either adding or
subtracting characters from a keyword
string transformString(const string &input, const string
&keyword, bool encrypt) {
    string result;
    for (size_t i = 0; i < input.size(); ++i) {
        if (encrypt) // If encrypting, add the keyword character to
the input character
            result += addChar(input[i], keyword[i % keyword.size()]);
        else // If decrypting, subtract the keyword character from
the input character
            result += subtractChar(input[i], keyword[i %
keyword.size()]);
    }
    return result;
}
```

Suggested Solution: (cont'd)

Third Part:

```
int main() {
    string input, keyword, operation;

    // Get the input from the user
    cin>> input;
    cin>> keyword;
    cin>> operation;

    // Check for invalid input
    if (input.size() > 20 || keyword.size() > 20 || operation.size() != 1 || operation[0] != 'E' && operation[0] != 'D') {
        cout << "Invalid Input" << endl;
        return 0;
    }
}
```

Suggested Solution: (cont'd)

Forth Part:

```
// Check each character in the input and keyword for validity
for (char c : input) {
    if (isInvalidChar(c)) {
        cout << "Invalid Input" << endl;
        return 0;
    }
}
for (char c : keyword) {
    if (isInvalidChar(c)) {
        cout << "Invalid Input" << endl;
        return 0;
    }
}
// Perform the encryption or decryption and print the result
cout << transformString(input, keyword, operation[0] == 'E') << endl;

return 0;
}
```



THANKS