Fundamental Computer Programming- C++ Lab(I)

Lab 4

Runs in a String

Week 4, Fall 2021

Rung-Bin Lin

International Bachelor Program in Informatics
College of Informatics
Yuan Ze University

Purposes

- Learn about characters and strings
- Learn how to use *for* loop
- Develop problem solving skills

String Type

- **#include <string>**
  - A string is stored as an array of characters.
  - An array is a continuous memory region that will store objects of the same type. The object's type can be *char*, *int*, *string*, *double*, etc. It has a sequence of locations. A location will hold exactly one object.
  - A string *constant* is double quoted by "", for example "astring".

A string is an array of characters.

char s1[] = "happy"; ←→ string s1("happy");

Length = 5 (not including end-of-string \0)

h	a	p	p	y	\0

char s2[] = " birthday"; ←→string s2(" birthday);

Length = 9

	b	i	r	t	h	d	a	y	\0

More on string and characters

- Each string is a character array. For example, if **aStr** is a string variable that stores a string "**He loves you.**", aStr[0] will contain 'H', aStr[1] contains 'e', aStr[2] contains ' ', aStr[3] contains 'l', aStr[4] contains 'o', etc.
- We can use a function **aStr.length()** to get the length of the string stored in aStr. It will return 13 for this example.
- Each character is stored in computers using ASCII. It is an integer. So we can have the following code.

```
int aDigit, anInt;
char aLetter;
aDigit = '9' – '0';   // aDigit will have a value of 9.
anInt = 'z' – 'a';    // anInt will have a value of 25.
aLetter = 'a' + 5;  // aLetter will be the letter f.
```

You may need use static_cast<char>(90) to convert an integer to a character.

Substring of a string

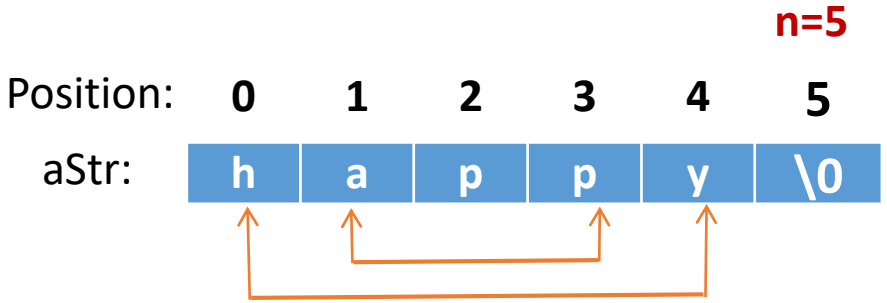
- **subStr** is a substring of a string **orgStr** if there exists an integer g such that subStr[i] == orgStr[g+i] for $0 \leq i < subStr.length(\quad)$.
- For example, given the following declaration,
  string orgStr = “He loves me.”;
  string subStr = “e loves m”;
  string subStr2 = “loves”;
  Here, both subStr and subStr2 are substrings of orgStr.

Manipulating a string

```
// Reversing a string
#include <iostream>
#include <string>
using namespace std;

Int main(){
    string aStr="abcdefg";
    char c;
    int strLen = aStr.length();
    for(int i=0; i<strLen/2; i++){
        c=aStr[i];
        aStr[i] = aStr[strLen-1-i];
        aStr[strLen-1-i] = c;
    }
    cout << aStr << endl;
    return 0;
}
```

- **Calculate string length**
  aStr.length();
- **Get access to a string element (i.e., a characters) at position i**
  aStr[i]
- **A string of length n is stored starting from position 0 through position n-1 in a character array.**
- **Reversing a string of length n can be done by exchanging the character at position n-1 with the one at position 0, the one at position n-2 with the one at position 1, etc. The code is shown on the left.**



Group Reverse

Group reversing a string means reversing a string by groups. For example consider a string:

"**TOBENU**MBERON**EWEMEE**TAGAIN**ANDAGA**INUNDE**RBLUEI**CPCSKY"

This string has length 48. We have divided into 8 groups of equal length and so the length of each group is 6. Now we can reverse each of these eight groups to get a new string:

"**UNEBOT**NOREBM**EEMEWE**NIAGAT**AGADNA**EDNUNI**IEULBR**YKSCPC"

Given the string and number of groups in it, your program will have to group reverse it.

Input

The input file contains at most 101 lines of inputs. Each line contains an integer G ($G < 10$) which denotes the number of groups followed by a string whose length is a multiple of G. The length of the string is not greater than 100. The string contains only alpha numerals. Input is terminated by a line containing a single zero.

Output

For each line of input produce one line of output which contains the group reversed string.

Example

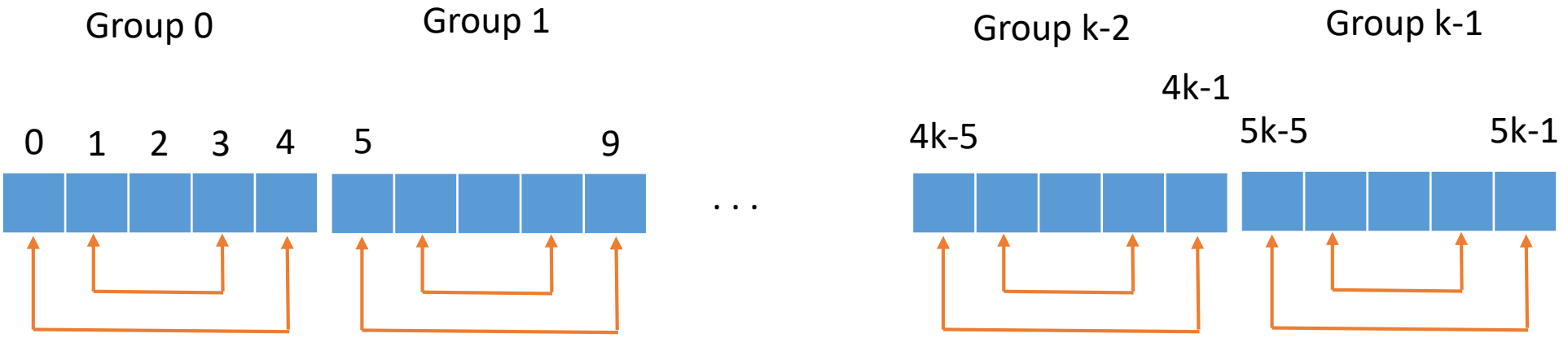
Sample Input

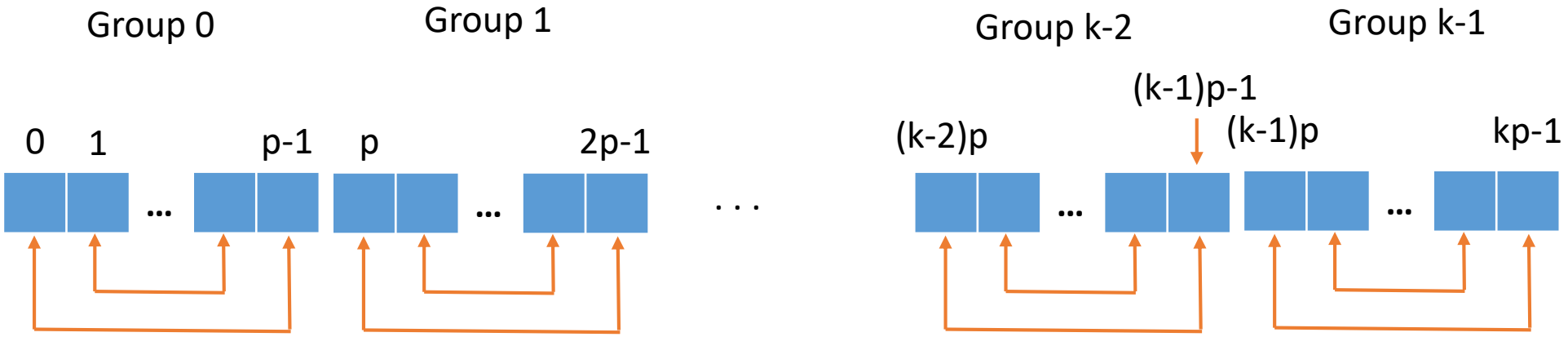
```
3 ABCEHSHSH
5 FAOETASINAHGRIONATWONOQAONARIO
0
```

Sample Output

```
CBASHEHSH
ATEOAFGHANISTANOIRAQONOWOIRANO
```

Group Reverse





Algorithm for Group Reverse

1. Input the number of group
2. Read a string
3. Calculate string length
4. For each of all the groups
   Reverse string
5. Print out the reversed string
6. Repeat (1) through (5) until end of input

Lab 4: Runs in a String

- Given a string without containing any whitespaces, find the number of runs, the length of the longest runs, and the number of longest runs. A run is a string's substring that contains only one type of symbol. A run is not contained in any other runs. The length of a run is the number of symbols in the run. For example, given a string "2Aaaa443333o#", "2" is a run of length 1; "Aaaa" is a run of length 4. The above string has 6 runs, the length of its longest run is 4, and the number of longest runs is 2. Here, the lower and upper cases of an alphabet letter are treated as the same symbol.
- Requirements:
  - Your program should use **cin** to read a whole string into the program before it find runs in the string.

Input & Output Formats

Input format

The first line gives the number of test cases. Each test case takes a line which contains a string.

Output Format

The output of a test case takes three lines as shown in the example on the right. The first line gives the number of runs, the second line gives the length of longest runs, and the third line gives the number of longest runs .

```
5
$####7435#@ASDddDDFff%
# Number of runs= 13
# Longest run length= 5
# Number of longest runs= 1
AaA
# Number of runs= 1
# Longest run length= 3
# Number of longest runs= 1
*
# Number of runs= 1
# Longest run length= 1
# Number of longest runs= 1
H655555eeeEeDdddD%%%%%%Yyyyyuytr
# Number of runs= 11
# Longest run length= 6
# Number of longest runs= 1
H655555eeeEeDdddD^^^^^YyyyYuYtrr
# Number of runs= 11
# Longest run length= 5
# Number of longest runs= 5
```

Follow All Requirements

- Input formats
- Output formats
- Coding styles
  - Avoiding using variables which do not have expressive power. That is, a variable name should carry the meaning of the matter in which the variable intends to represent.

If you don’t follow the requirements, up to 30% of the points for your lab will be deduced.

Rules for Program Submission

- Put all the relevant files in the same folder.
- Name your folder SID_LabX, where ID is your student ID number and X is the number assigned to the lab. If a lab has N parts, N>1, then create N sub-folders with their names SID_LabX_N in the the folder SID_LabX.
  - For example, for Lab 2 with only one part and with student ID number 1041544, the name of the folder must be S1041544_Lab2. N is omitted if there is only one part.
  - Another example, similar to the above but Lab 2 has two parts. Then, you have to create a folder S1041544_Lab2 and two sub-folders S1041544_Lab2_1 and S1041544_Lab2_2
- Compress the folder into a file named SID_LabX.zip, for example, S1041533_Lab2.zip. Then, submit the compressed file
- If you violate this rule, your lab will not be graded. If graded other penalty will be applied.