

String + Character Processing VII:

∇ stands for a space character.

String Comparison:

All comparison operators can be used to compare strings in Pascal.

The (=) operator check if two strings are equal to each other.

Similarly, the (< >) operator check whether there are different characters.

For the (>), (<), (>=), (<=) operator .

To perform comparison, we compare character by character of two strings.

For example, 'abbcc' and 'abeef'

'a' = 'a'

'b' = 'b'

'b' is less than 'e'

So 'abbcc' is less than 'abeef'

Another example, 'abb' and 'abbccc'

'a' = 'a'

'b' = 'b'

'b' = 'b'

Since there is no more character in 'abb' , it is less than 'abbccc'.

Remarks:

Null-String is a string that consists of nothing, *which is less than or equal to any string.*

Always remember that you can't compare two values of different types.

Which of the following string is greater?

(Every string are separated by a comma, if there is any syntax error, write them down)

- 1) '20', '1000', '13456'
- 2) 'Abbcxc', 'aBBCXC', '12ddd'
- 3) '∇∇∇', '∇', '∇∇'
- 4) ", ""', '∇∇'
- 5) ""', ""', ""'
- 6) 'It"sveryhot.', 'It"sveryhot', 'it"sVeryhot'
- 7) '13458783E+01', '233E+06', '1.99999E+99'
- 8) '0.1224', '1.343', '-13.42'
- 9) 44, 044, 142
- 10) FALSE, TRUE
- 11) 'FALSE', '20', 'TRUE'

String + Character Processing VIII:

Revision Exercise: ($0 \leq N \leq 10^9$) ($2 \leq B, P \leq 16$)

- 1) Write a program to convert a denary number N to binary number.
- 2) Write a program to convert a denary number N to octal number.
- 3) Write a program to convert a denary number N to hexadecimal number.
- 4) Write a program to convert a given string to UPPER CASE.
- 5) Write a program to convert a given string to LOWER CASE.
- 6) Write a program to convert a hexadecimal number N to denary number.
- 7) Write a program to convert a denary number N to B-based number.
- 8) Finally, write a program to convert a P-based number to B-based number.

Answer: *Read after you finish the above exercises or if you find any difficulties.*

```
program q1;
const d='0123456789ABCDEF';
    b=2;
var n:longint; s:string;
begin
readln(n); s:="";
repeat
s:=copy(d,n mod b+1,1)+s;
n:=n div b;
until n=0;
writeln(s);
end.
```

```
program q2;
const d='0123456789ABCDEF';
    b=8;
var n:longint; s:string;
begin
readln(n); s:="";
repeat
s:=copy(d,n mod b + 1,1)+s;
n:=n div b;
until n=0;
writeln(s);
end.
```

```
program q3;
const d='0123456789ABCDEF';
    b=16;
var n:longint; s:string;
begin
readln(n); s:="";
repeat
s:=copy(d,n mod b + 1,1)+s;
n:=n div b;
until n=0;
writeln(s);
end.
```

```
program q4;
var n:longint; s:string;
begin
readln(s);
for n:=1 to length(s) do
if (s[n]<='z') and (s[n]>='a') then
s[n]:=chr(ord(s[n])-ord('a')+ord('A'));
writeln(s);
end.
```

```
program q5;
var n:longint; s:string;
begin
readln(s);
for n:=1 to length(s) do
if (s[n]<='Z') and (s[n]>='A') then
s[n]:=chr(ord(s[n])-ord('A')+ord('a'));
writeln(s);
end.
```

```
program q7;
const d='0123456789ABCDEF';
var n,b:longint; s:string;
begin
readln(n,b); s:="";
repeat
s:=copy(d,n mod b + 1,1)+s;
n:=n div b; until n=0;
writeln(s);end.
```

```
program q6;
const d='0123456789ABCDEF';
    p=16;
var n,i:longint; s:string;
begin
readln(s); n:=0;
for n:=1 to length(s) do
if (s[n]<='z') and (s[n]>='a') then
s[n]:=chr(ord(s[n])-ord('a')+ord('A'));
for i:=1 to length(s) do
n:=p*n+(pos(s[i],d)-1);
writeln(n);
end.
{d is not a variable , so you can't use d[n] }
```

```
program q8;
const d='0123456789ABCDEF';
var n,i:longint; s:string;
begin
readln(s); readln(p,b); n:=0;
for n:=1 to length(s) do
if (s[n]<='z') and (s[n]>='a') then
s[n]:=chr(ord(s[n])-ord('a')+ord('A'));
for i:=1 to length(s) do n:=p*n+(pos(s[i],d)-1);
s:=""; repeat
s:=copy(d,n mod b+1,1)+s;
n:=n div b; until n=0;
writeln(s);
end.
```

String + Character Processing IX:

We will move slowly to Free Pascal from now on.
Please be familiar with FP's data type.

Strings Function II:

o : <ordinary type value> ; **c** : char; **ss** : shortstring; **as**:ansistring;

Function	Meaning
pred(o)	Return the predecessor s. {The previous element}
succ(o)	Return the successor of s. {The next element}
Uppcase(c)	Return the upper case character of c.
Uppcase(ss) *	Return the upper case shortstring of ss.
Uppcase(as) *	Return the upper case ansistring of as.
Lowercase(c) *	Return the lower case character of c..
Lowercase(ss) *	Return the lower case shortstring of ss.
Lowercase(as) *	Return the lower case ansistring of as.

*** : Only exists in Free Pascal.**

Example:

Pred('s') return 'r'.

Succ('s') return 't'.

Uppcase('s') return 'S'.

Uppcase('saa22') return 'SAA22'.

Lowercase('S') return 's'.

Lowercase('SAA22') return 'saa22'.

Strings Procedure II:

len : longint ; **c** : char; **ss** : shortstring; **as**:ansistring;

Procedure	Meaning
Setlength(ss,len) *	Set the length of shortstring ss to len.
Setlength(as,len) *	Set the length of ansistring as to len.

Although we change the value of s[4] , s[5] , the compiler still treats s as 'abc' instead of 'abcdefg' , because we didn't change the length of the string.

```
var s:string; begin
s:='abc';
s[4]:='d';
s[5]:='e';
writeln(s); {Output 'abc'}
end.
```

Notice the use of setlength.

```
var s:string; c:char;
begin
s:='abc'; s[4]:='d'; s[5]:='e';
setlength(s,5);
writeln(s); {Output 'abcde'}
end.
```

In TPW, we can use s[0]:=chr(5) to set the length of s to 5.

String + Character Processing X:

Supplementary to string, character: (*You may ignore this part...*)

Besides using the function `chr`, we can also get the corresponding character of a given ASCII code by using the syntax `#xxx`.

`#XXX` { XXX must be constant integer , $0 \leq XXX \leq 255$ }

For example, `#97` is exactly the same as `'a'`.

Also, for `#1`, `#2`, ..., `#26`, we can use another syntax

`^Y` { Y must be a character }

Therefore, `#1` is equal to `^a` and `chr(1)`. `#2` is equal to `^b` and `chr(2)`, and so on.

Practical Exercises:

- 1) Write a program to find the **largest repeating prefix** of a string `S`. { $\text{length}(s) \leq 255$ }
(Read LPR.txt for description of **largest repeating prefix**.)

Sample: (*bold and italic stands for user input*)

<i>BABATBABATABAT</i> BABAT	<i>5554555412233</i> 5554
<i>ABCACC</i> No repeating prefix.	<i>ABCDEGABCBCDEFABCDEGABCBCDEFABC_NEE</i> ABCDEGABCBCDEF

- 2) Output the **Roman Numeral Representation** of a integer `N`. { $1 \leq N \leq 3999$ }

Sample: (*bold and italic stands for user input*)

<i>12</i> XII	<i>1999</i> MCMXCIX
<i>3333</i> MMMCCCXXXIII	<i>2047</i> MMXLVII

- 3) Write a program to find out number of occurrences of each character of a string.
- 4) Find out all pair of (a,b,p) such that $a^b + b^a = p$, where a,b,p are prime numbers.

Exercise for String Processing:

(Hand in on 13-Jul)

Exercise I:

1999hje.doc	36 , 44*
1999hse.doc	34 , 35
2000hje.doc	22 – 23 , 27 , 40
2000hse.doc	7 , 12 , 20
2002hje.doc	4 , 13 , 18
2003hje.doc	A11 , A19

Exercise II:

1999hje.doc	34
1999hse.doc	14 , 22 – 23* , 30
2000hse.doc	10
2002hje.doc	9 , 31
2004hje.doc	B1 , B4 , B11

Exercise III: (Difficult) (Hand in on 18-Jul)

Find the **denary representation** of a given **Roman Numeral Representation N**.

{1 ≤ N ≤ 3999}

<i>XII</i> 12

<i>MCMXCIX</i> 1999

<i>MMMCCCXXXIII</i> 3333

<i>MMXLVII</i> 2047

(Send the source code to me when you finish and I will check if your program is correct.)

Exercise IV: (Difficult)

Write a program to solve **Number Spirals.txt**

Exercise V: (Very difficult)

Write 2 programs to output the following patterns.

Input:5 Output: 11111 12221 12321 12221 11111

Input:3 Output: 111 121 111

Input:5 Output: 12321 22322 33333 22322 12321

Input:3 Output: 121 222 121

Input number is ≤ 9 and odd.