分数构成

平时成绩构成

```
* 15% Labs
* 10% Weekly Programming Tests
* 15% Assignment 1 — due week 7
* 15% Assignment 2 — due week 10
* 45% Final Exam
```

考试注意

```
# 老师发了邮件，闭卷
# 不能登陆课件，自己存一份
# 初步怀疑，老师可能会全屏，禁止复制
# 5000的问， 不要给别人代码
```

考试通过

```
* score 50/100 overall
* score 18/45 on final exam
```

考试时间

```
* Run under same conditions as Weekly Tests
* Except 3 hours and some question may not be coding
* Saturday 22 August 13:00 — 16:00
* Exam will be released on class web site at 12:50
* Questions during exam can be sent to cs2041.exam@cse.unsw.edu.au
```

考试的要求

```
* You are not permitted to communicate (email, phone, message, talk, . . . )
to anyone but COMP2041|COMP9044 staff during exam
* You are not permitted to get help from anyone but COMP2041|COMP9044 staff
during the exam.
* This is a closed book exam: you are not permitted to access papers, books,
files on your computer or the internet
* You are permitted to access the exam web pages on the class web site
* You are permitted to access the online language cheatsheets & documentation
on the class web site
* Deliberate violation of exam conditions will be treated as serious
misconduct
```

考试的形式

```
* 12-15 questions
* Each questions answered in a separate file.
* Some questions will ask you to write shell.
    or perhaps a shell pipeline
* Some questions will ask you to write Perl.
* Answers will be submitted with give.
* Questions not equal difficulty
* Questions may not be worth equal marks
* File will be submitted with give.
```

考试代码的说明

```
• Questions will usually include examples.
• You may or may not be given starting code.
• You may or may not be given test data or other files
• 1 or more autotests may be available on submission.
• Passing autotests does not guarantee any marks. Do your own testing.
• There may be no submission tests for some questions.
• It is not sufficient to match any supplied examples
```

考试评分标准

```
• Answers will be run through automatic marking software.
    • Please follow the input/output format shown exactly.
    • Please make your program behave exactly as specified.
• All answers are hand marked, guided by automarking.
    • No marks awarded for style or comments . . .
    • But use decent formatting so the marker can read the program!
    • Comments only necessary to tell the marker something.
• Minor errors will result in only a small penalty.
    • e.g. an answer correct except for a missing semi-colon would receive
almost full marks.
• No marks will given unless an answer contains a substantial part of a
solution (> 33%).
• No marks just for starting a question and writing some code
```

shell 命令

推荐

```
1.如果是grep，尽量用egrep
    echo 'a a the' | grep '[a|the]'

    echo 'a a the' | egrep '(a|the)'
    echo 'a a the' | egrep 'a|the'
    echo 'a a the' | egrep '(a|the)'
```

标准输入

```
./a.sh data1
./a.sh < data1
```

grep

```
-i 或 --ignore-case : 忽略字符大小写的差别。
-o 或 --only-matching : 只显示匹配PATTERN 部分。
-v 或 --revert-match : 显示不包含匹配文本的所有行。
```

cat

```
-n 或 --number：由 1 开始对所有输出的行数编号。
-s 或 --squeeze-blank：当遇到有连续两行以上的空白行，就代换为一行的空白行。
```

wc(word count)

```
-c或--bytes或--chars 只显示Bytes数。
-l或--lines 只显示行数。
-w或--words 只显示字数。
```

tr

```
-c, --complement：反选设定字符。也就是符合 SET1 的部份不做处理，不符合的剩余部份才进行转换
-d, --delete：删除指令字符
-s, --squeeze-repeats：缩减连续重复的字符成指定的单个字符
```

head/tail

```
-n option changes number of lines head/tail prints.
```

egrep

```
-i ignore upper/lower-case difference in matching
-v only display lines that do not match the pattern
-w only match pattern if it makes a complete word
```

Regular Expressions

```
[^a-e]  非a-e开头的字符集
^[abc]  以abc开头的行
cat$    以cat结尾的行

p*      0个或者多个
p+      1个或者多个
p?      0个或者1个

a.c     必须有一个字符在a和c之间，.叫做占位符
ab*c    a和c之间有0到多个b

[a|the]   a或者the
[a-z]   匹配所有的小写字母
[0-9]   匹配所有的数字
[\d+]   digit 匹配
```

cut

```
-b ： 以字节为单位进行分割
-c ： 以字符为单位进行分割。
-d ： 自定义分隔符，默认为制表符(\t)。
-f ： 与-d一起使用，指定显示哪个区域。

cut -f1 data.txt    取第一列
cut -f1-3 data.txt  第一列到第三列

cut -f1,4 data.txt  取第一列，第四列
cut -f4- data.txt   第四列之后的所有的列

cut -d'|' -f1-3 data.txt    '|'分割后，取第一列到第三列
cut -c1-5 data.txt          取第一个到第5个字符
```

sort

```
-b 忽略每行前面开始出的空格字符。
-d 排序时，处理英文字母、数字及空格字符外，忽略其他的字符。
-f 排序时，将小写字母视为大写字母。
-n 依照数值的大小排序。
-r 以相反的顺序来排序。
-u 意味着是唯一的(unique)，输出的结果是去完重了的。

sort -nr -k3 data
```

uniq remove or count duplicates

```
-c或--count 在每列旁边显示该行重复出现的次数。
-d或--repeated 仅显示重复出现的行列。
-u或--unique 仅显示出一次的行列
```

sed

```
LineNo  selects the specified line
StartLineNo,EndLineNo
        selects all lines between specified line numbers
/RegExp/
        selects all lines that match RegExp
/RegExp1/,/RegExp2/
        selects all lines between lines matching reg exps

sed -n -e '1,10p' < file
sed -n -e '81,100p' < file

sed -n -e '/xyz/p' < file
```

```
sed -e '/[xyz|abc]/d' < file

sed -e 's/[^:]*://' datafile
```

```
#
sed 's/xyz//g';
# 如果不加[]表示xyz是一个整体，加上[]表示里面的内容是单个自负

# 不加中括号，匹配abc或者xyz
echo 'xab abc' | egrep -o 'abc|xyz'
# 加上中括号，表示中括号里面的每一个字符
echo 'xab abc' | egrep -o '[abc|xyz]'
```

find

```
find /home/jas/web -name '*.html'
```

basename

```
file=/home/jas/web/a.html
file_name=`basename $file`
```

Shell

command line args

```
$0 the name of the command
$1 the first command-line argument
$2 the second command-line argument
$3 the third command-line argument
$# count of command-line arguments
$* all of the command-line arguments (together) $@ all of the command-line
arguments (separately) $? exit status of the most recent command
$$ process ID of this shell
```

输入输出

```
read,echo
```

变量定义

```
没有空格
$ x=5
$ y="6"
$ z=abc
$ echo $(( $x + $y ))
```

debug

```
set -x shows each command after transformation
```

Quoting

```
single-quote (’)    grouping, turns off all transformations
double-quote (")    grouping, no transformations except $ and ‘
backquote (‘)  no grouping, capture command results
```

test

```
bool expression
# 重点去强调的, perl反着的
string comparison ( = != )
numeric comparison ( -eq -ne -lt -gt ) checksonfiles (-f -x -r)
boolean operators ( -a -o ! )
test "$msg" = "Hello"
test "$x" -gt "$y"
test "$x" -ge 10 -a "$x" -le 20
test -r xyz -a -d xyz

if test ! -r "$f"   # is the arg readable?
then
    echo "No such file: $f"
else
```

```
-e 文件名  如果文件存在则为真
-r 文件名  如果文件存在且可读则为真
-w 文件名  如果文件存在且可写则为真
-x 文件名  如果文件存在且可执行则为真
-s 文件名  如果文件存在且至少有一个字符则为真
-d 文件名  如果文件存在且为目录则为真
-f 文件名  如果文件存在且为普通文件则为真
-c 文件名  如果文件存在且为字符型特殊文件则为真
-b 文件名  如果文件存在且为块特殊文件则为真
```

if 语句

```
if testList{1} then
    commandList{1}
elif testList{2} then
    commandList{2} ...
else
    commandList{n}
fi
```

```
if grep "^$user" /etc/passwd > /dev/null then
    # do something if they do exist ...
else
    echo "$0: $user does not exist"
fi

if diff -q a.txt a1.txt > /def/null
then
    echo 'same'
else
    echo 'difference'
fi
```

for 语句

```
# 特别注意一下的
sum=0
for n in "$@"
do
    sum=‘expr $sum + "$n"‘
done

echo $(( $sum + "$n" ))
```

```
# 比较两个文件夹下的文件是否相等
# 下面的这种写法，是可以处理文件名中有空格的
for file in `ls $dir1`
do
done

for ls_file1 in $dir1/*
do
    file_name=`basename "$ls_file1"`
    if test -e "$dir2/$file_name"
    then
```

```
        if diff -i -w "$dir1/$file_name" "$dir2/$file_name" >/dev/null
        then
            echo "$file_name"
        fi
    fi
done
```

while 语句

```
while true
do
    if diff -q a.txt a1.txt > /def/null
    then
        echo 'same'
    else
        exit
    fi
done
```

Perl

变量定义

```
$x = '123';
$y = "123 ";
$z = 123;
$i = $x + 1;

$a .= "abc"
```

逻辑运算

```
And     x && y      x and y
Or      x || y      x or y
Not     ! x         not x

Numeric: == != < <= > >= <=>
String: eq ne lt le gt ge cmp
```

文件读取的方式

```
$line =~ s/\s*$//g;
```

```
while ($line = <>) {
    # 仅过滤换行符
    chomp $line;
    print $line;
}

open my $files, '-|', "ls $d";
while (<$files>) {
    chomp;
    @fields = split;
    print "Next file is $fields
}

-r, -w, -x      file is readable, writeable, executable
-e, -z, -s      file exists, has zero size, has non-zero size
-f, -d, -l      file is a plain file,directory, sym link
```

特殊变量

```
$_ default input and pattern match
@ARGV list (array) of command line arguments
$0 name of file containing executing Perl script (cf. shell)
$i matching string for ith regexp in pattern
$! last error from system call such as open
$. line number for input file stream
$/ line separator, none if undefined
$$ process number of executing Perl script (cf. shell)
%ENV lookup table of environment variables
```

定义数组

```
@a = ("first string", "2nd string", 123);
@a = ();

$n=@a;

@numbers = (4, 12, 5, 7, 2, 9);
($a, $b, $c, $d) = @numbers;

($x, $y) = ($y, $x);
```

循环

```
@nums = (23,95, 33, 42, 17, 87, 10, 20);
$sum = 0;
for ($i = 0; $i < @nums; $i++) {
    $sum += $nums[$i];
}
$sum = 0;
foreach $num (@nums) {
    sum += $num;
}
```

list operations

```
# sort compare function
# 先按照长度，再按照字典顺序排序
{$a <=> $b}

sort,reverse,push,pop,shift,unshift
split,join

split 支持正则分割
# TODO
```

字典的调用

```
while (($key,$val) = each %myHash) {
    print "($key, $val)\n";
}

foreach $x (keys %g) {
    print "$x    => $g{$x}\n";
}

foreach $val (values %myHash) {
    print "(?, $val)\n";
}

%two_dict = ();

$two_dict{"1"}{"b"} = 1;

foreach $x (keys %{$two_dict{"1"}}) {

}
```

Regular Expressions

```
$name =~ /[0-9]/
$name =~ s/Mc/Mac/;
$string =~ tr/a-z/A-Z/;

$string =~ tr/A-Z/a-z/;

\d matches any digit, i.e. [0-9]
\D matches any non-digit, i.e. [^0-9]
\w matches any "word" char, i.e. [a-zA-Z_0-9]
\W matches any non "word" char, i.e. [^a-zA-Z_0-9]
\s matches any whitespace, i.e. [ \t\n\r\f]
\S matches any non-whitespace, i.e. [^ \t\n\r\f]

\b matches at a word boundary
\B matches except at a word boundary
patt*   matches 0 or more occurences of patt
patt+   matches 1 or more occurences of patt
patt ?  matches 0 or 1 occurence of patt

patt {n,m}  matches between n and m occurences of patt

$pattern = "ab+";
$replace = "Yod";
$text = "abba";
$text =~ s/$pattern/$replace/;

$string = "-5==10zzz200_";
@numbers = $string =~ /\d+/g;
print join(",", @numbers), "\n";
# prints 5,10,200
```

```
$string = "Bradley, Marion Zimmer";
($family_name, $given_name) = $string =~ /([^,]*), (\S+)/;

$1, $2

print "$given_name $family_name\n";
# prints Marion Bradley
```

方法调用

```
sub mySub {
    @args = @_;
    print "I got ",@#args+1," args\n";
    print "They are (@args)\n";
}
```

```
sub f {
    my ($x, $y, $z) = @_;
    my $result;
    ...
    return $result;
}

sub good {
    my ($x, $y, @list) = @_;
}

# 数组传递的地址引用
sub mypush {
    my ($array_ref,@elements) = @_;
    if (@elements) {
        @$array_ref = (@$array_ref, @elements);
    } else {
        @$array_ref = (@$array_ref, $_);
    }
}

mypush(\@array, $x);
```

sort 方法使用（第七周）的代码全是sort

```
foreach $c (sort {$n_taking{$a} <=> $n_taking{$b}} keys %n_taking) {
    printf "%5.1f%% of %s students take %s %s\n",
        100*$n_taking{$c}/$n_students, $course, $c, $course_name{$c};
}

sub compare_date {
    my ($day1,$month1,$year1) = split /\D+/, $a;
    my ($day2,$month2,$year2) = split /\D+/, $b;
    return $year1 <=> $year2 || $month1 <=> $month2 || $day1 <=> $day2;
}

@sorted_dates = sort compare_date @random_dates;
```

常用正则

```
匹配数字：-?\d+(\.\d+)?      正负以及小数
匹配数字：^[0-9]+(\.[0-9]*)?     整数 + 小数
```

常用的类库

```
use File::Compare
use File::Copy

use experimental 'smartmatch';
# 判断数组存在不存在
next if ($line ~~ @results);
# 判断字典存在不存在
%results =();
next if exists $results{$line};

use List::Util qw(min max);
```

shell的bool 表达式：

```
# 0表示的是真的
# 1表示的是假的
if test 1 -gt 2
then
    echo $?
    echo '0'
else
    echo $?
    echo '1'
fi
```

```
\s 包括什么东西？空格，\t,\r,\n

split(/^\d+/,$line);

split(/\d+/,$line);
```

```
[1-9][0-9]*\.[0-9]+
```

命令

```
mv 移动文件
read
chmod
cp
rm
cat
mkdir
date
```