CH 5: Loops and Files

P1. Write a program to generate 4 random numbers and output each generated number.

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

int main()
{
	short seed;
	int randNum;

	const int MAX = 20,
		  MIN = 5;

	seed = time(0);
	srand(seed);

	randNum = rand() % (MAX - MIN) + MIN;
	cout << "random number " << randNum << endl;

	randNum = rand() % (MAX - MIN) + MIN;
	cout << "random number " << randNum << endl;

	randNum = rand() % (MAX - MIN) + MIN;
	cout << "random number " << randNum << endl;

	randNum = rand() % (MAX - MIN) + MIN;
	cout << "random number " << randNum << endl;


	return 0;
}
```

VS

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


int main()
{
	short seed;
	int randNum;

	const int MAX = 20,
		  MIN = 5;

	seed = time(0);
	srand(seed);

	for (int i = 0; i < 4; i = i + 1)
	{
		randNum = rand() % (MAX - MIN) + MIN;
		cout << "random number " << randNum << endl;
	}

	return 0;
}
```

5.1 Increment and Decrement Operators.

Recall:

a += 5; equivalent to a = a + 5;

+= is a binary operator

increment and decrement operators are unary operators.

	Postfix mode	Prefix mode	Equivalent to	Value of a (assume a = 10;)
increment	a++;	++a;	a = a + 1;	10 + 1 = 11
decrement	a--;	--a;	a = a – 1;	10 – 1 = 9

Predict the output (increment operator) :

```
num = 8;
cout << num << endl;

num ++ ;
cout << num << endl;

++ num;
cout << num << endl;
```

```
.....


.....

.....
```

Predict the output (decrement operator):

```
num = 13;
cout << num << endl;

num --;
cout << num << endl;

-- num;
cout << num << endl;
```

```
.....


.....

.....
```

Predict the output (combined statements with increment operator):

Note:

```
cout << num ++ << endl;
```

equivalent to

```
cout << num << endl;
num = num + 1;
```

```
cout << ++ num << endl;
```

equivalent to

```
num = num + 1;
cout << num << endl;
```

```
num = 7;
cout << num << endl;

cout << num ++ << endl; // displays 7 and
                        // adds 1 to num

cout << ++ num << endl; // adds 1 to num
                        // and displays 9
```

```
.....................................

.....................................

.....................................
```

Predict the output (combined statements with decrement operator):

Note:

```
cout << num -- << endl;
```

equivalent to

```
cout << num << endl;
num = num - 1;
```

```
cout << -- num << endl;
```

equivalent to

```
num = num - 1;
cout << num << endl;
```

```
num = 8;
cout << num << endl;

cout << num -- << endl; // displays 8 and subtract 1 from num

cout << -- num << endl; // subtracts 1 from num and displays 6
```

a = 2, b = 5, c = 6;	Equivalent to	output
b = a++; cout << a << " " << b << endl;	b = a ; a = a +1;	
c = a * b++; cout << a << " " << b << " " << c;	c = a * b; b = b +1;	
bool x = (c++ > 6); cout << c << " " << x;	bool x = (c > 6); c = c + 1;	
(a * b) ++	(a * b) = (a * b) + 1	

a = 2, b = 5, c = 6;	Equivalent to	output
b = ++ a; cout << a << " " << b << endl;	a = a +1; b = a ;	
c = a * ++ b; cout << a << " " << b << " " << c;	b = b +1; c = a * b;	
bool x = (++ c > 6); cout << c << " " << x;	c = c + 1; bool x = (c > 6);	
++ (a * b)	(a * b) = (a * b) + 1	

x = 2, y = 5, z = 6;	Equivalent to	output
y = x--; cout << x << " " << y<< endl;	y = x ; x = x +1;	
z = x * y--; cout << x << " " << y << " " << z;	z = x * y; y = y +1;	
bool x = (c-- > 6); cout << c << " " << x;	bool x = (z > 6); z = z + 1;	
(x * y) --	(x * y) = (x * y) + 1	

LOOPS

Three types of loops:

- *for* loop (sec 5.6)
- *while* loop (sec 5.2)
- *do-while* loop (sec 5.5)

a typical loop has 4 major parts:

- initialization
- test/ condition
- update
- statements to execute

For loop

```
for (initialization; test; update)
{
     statements
}
```

```
for (int i = 0; i < 10; i ++)
{
    randNum = rand() % (MAX - MIN) + MIN;
    cout << "random number " << randNum
         << endl;
}
```

Notes:

- no semicolon after the parenthesis.
- Each part in the parenthesis is separated by a semicolon.

while loop

```
initialization
while (test)
{
   statements
   update
}
```

```
i = 0;
while (i < 10)
{
    randNum = rand() % (MAX - MIN) + MIN;
    cout << "random number " << randNum
         << endl;
    i ++;
}
Note: assume variable i has been defined
at the top of the main function.
```

Notes:

- no semicolon after the test (aka expression) in parenthesis.
- The statements that in braces are called the body.
- Each repetition is called one iteration.
- Variable *i* is called the loop control variable.

do-while loop

```
initialization
do
{
     Statements;
     Update;
} while (test);
```

```
i = 0;
do
{
    randNum = rand() % (MAX - MIN) + MIN;
    cout << "random number " << randNum
         << endl;
    i ++ ;
}
while (i < 10);

Note: assume variable i has been defined
at the top of the main function;
```

Notes:

- There is a semicolon after the test (aka expression) in parenthesis.

for	**while**	**do-while**
Ideal for performing a known number of iterations.	The loop will never execute if the expression is false to start with. To make it executes the first time, relevant data must be initialize such that the expression results out as true.	Ideal for cases you must execute at least once.
Pretest loop	Pretest loop	Posttest loop

P2. Write a loop to output your name 5 times.

for loop

while loop

do-while loop

Conclusion: for loop is easier to write and understand for this problem.

P3. Write a loop to output numbers from 0 to 4.

for loop

while loop

do-while loop

P4. Write a loop to output numbers from 1 to 10.

for loop

while loop

do-while loop

P5. Write a loop to output even numbers from 1 to 10.

for loop

while loop

do-while loop

P6. Write a loop to "Count Reverse" from 5 to 0 using a for loop; output the values.

for loop

while loop

do-while loop

P7. Write a program to find the square root of a user entered number. (note that square root can be found only of a non-negative number.)

while loop

Note that number is read at least once. So, do-while loop can be utilized.

do-while loop

P8. Write a program to find the result of division. The program takes the two numbers from the user. (note that the denominator cannot be zero.)

while loop

Note that number is read at least once. So, do-while loop can be utilized.

do-while loop