C++ 指针实验题目

说明

指针是一个变量，其值为另一个变量的地址，即，内存位置的直接地址。就像其他变量或常量一样，您必须在使用指针存储其他变量地址之前，对其进行声明。指针变量声明的一般形式为：

```
type *var-name;
```

定义指针

在这里，type 是指针的基类型，它必须是一个有效的 C++ 数据类型，`var-name` 是指针变量的名称。用来声明指针的星号 `*` 与乘法中使用的星号是相同的。但是，在这个语句中，星号是用来指定一个变量是指针。以下是有效的指针声明：

```
int    *ip;    /* 一个整型的指针 */
double *dp;    /* 一个 double 型的指针 */
float  *fp;    /* 一个浮点型的指针 */
char   *ch;    /* 一个字符型的指针 */
```

所有指针的值的实际数据类型，不管是整型、浮点型、字符型，还是其他的数据类型，都是一样的，都是一个代表内存地址的长的十六进制数。不同数据类型的指针之间唯一的不同是，指针所指向的变量或常量的数据类型不同。

题目1 最大值与最小值

题目描述

采用指针法，输出十个整形数中最大值和最小值

输入输出格式

输入格式

10个整形数

输出格式

最大值和最小值

输入输出样例

输入样例

```
2 6 3 8 1 5 7 0 4 9
```

输出样例

```
9 0
```

题目2 字符串倒序输出

题目描述

编写主程序，将输入字符串反序输出

输入输出格式

输入格式

一串字符串

输出格式

倒序输出字符串

输入输出样例

输入样例

```
ABCDEFGHIJK
```

输出样例

```
KJIHGFEDCBA
```

题目3 冒泡排序

题目描述

用指针编写一个对整型数组进行冒泡排序函数。冒泡排序是指将相邻的元素进行比较，如果不符合所要求的顺序，则交换这两个元素；对整个数列中所有的元素反复运用上法，直到所有的元素都排好序为止。

输入输出格式

输入格式

数组大小 输入的数

输出格式

输出数据

输入输出样例

输入样例

```
5
503 87 512 61 908
```

输出样例

```
61 87 503 512 908
```

题目4 分节输出

题目描述

编写程序，将某一个输入的位数不确定的正整数按照标准的三位分节格式输出，例如，当用户输入82668634时，程序应该输出82,668,634

输入输出格式

输入格式

82668634

输出格式

按照，分割的值

输入输出样例

输入样例

```
82668634
```

输出样例

```
82,668,634
```

输入样例

```
1
```

输出样例

```
1
```

输入样例

```
4111
```

输出样例

```
4,111
```

题目5

C++空指针

在变量声明的时候，如果没有确切的地址可以赋值，为指针变量赋一个 NULL 值是一个良好的编程习惯。赋为 NULL 值的指针被称为空指针。

NULL 指针是一个定义在标准库中的值为零的常量。请看下面的程序：

```
#include <iostream>

using namespace std;

int main ()
{
   int  *ptr = NULL;

   cout << "ptr 的值是 " << ptr ;

   return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
ptr 的值是 0
```

在大多数的操作系统上，程序不允许访问地址为 0 的内存，因为该内存是操作系统保留的。然而，内存地址 0 有特别重要的意义，它表明该指针不指向一个可访问的内存位置。但按照惯例，如果指针包含空值（零值），则假定它不指向任何东西。

如需检查一个空指针，可以使用 if 语句，如下所示：

```
if(ptr)     /* 如果 ptr 非空，则完成 */
if(!ptr)    /* 如果 ptr 为空，则完成 */
```

因此，如果所有未使用的指针都被赋予空值，同时避免使用空指针，就可以防止误用一个未初始化的指针。很多时候，未初始化的变量存有一些垃圾值，导致程序难以调试。

题目6

返回指针

我们已经了解了 C++ 中如何从函数返回数组，类似地，C++ 允许从函数返回指针。为了做到这点，必须声明一个返回指针的函数。

```
int * myFunction()
{
.
.
.
}
```

另外，C++ 不支持在函数外返回局部变量的地址，除非定义局部变量为 static 变量。

现在，让我们来看下面的函数，它会生成 10 个随机数，并使用表示指针的数组名（即第一个数组元素的地址）来返回它们，具体如下：

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

// 要生成和返回随机数的函数
int * getRandom( )
{
  static int  r[10];

  // 设置种子
  srand( (unsigned)time( NULL ) );
  for (int i = 0; i < 10; ++i)
  {
    r[i] = rand();
    cout << r[i] << endl;
  }

  return r;
}

// 要调用上面定义函数的主函数
int main ()
{
   // 一个指向整数的指针
   int *p;

   p = getRandom();
   for ( int i = 0; i < 10; i++ )
   {
       cout << "*(p + " << i << ") : ";
       cout << *(p + i) << endl;
   }

   return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
624723190
1468735695
807113585
976495677
613357504
1377296355
1530315259
1778906708
1820354158
667126415
*(p + 0) : 624723190
*(p + 1) : 1468735695
*(p + 2) : 807113585
*(p + 3) : 976495677
*(p + 4) : 613357504
*(p + 5) : 1377296355
*(p + 6) : 1530315259
*(p + 7) : 1778906708
*(p + 8) : 1820354158
*(p + 9) : 667126415
```