```
/* Thanks to former CS 31 TA Kung-Hua Chang for the set of practice
problems and solutions.
Ref: Practice Problems for C++ Beginners: Moving Beyond the Basics,
by Dr. Kung-Hua Chang. */
```

1. What is the output of the program?

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

int main() {
        int *p = new int;
        * p = 100;

        cout << p << endl;
        cout << *p << endl;

        *p = *p + 7; // What about p = p + 7?
        cout << p << endl;
        cout << *p << endl;

        delete p;
}
```

Solution:

```
00389730
100
00389730
107
```

Note: You should see a different address other than 00389730. As for p = p + 7, this adds 7 to the memory address stored in p instead of the content of the address stored in p.

2. If the following program doesn't compile, why not? If it does compile, what is the output when it is run?

```
#include <iostream>
using namespace std;
int main()
{
        const double pi= 3.14;
        double *p = & pi;

        *p = 2;
        cout << *p << endl;
}
```

Solution:

Compilation error because a non-constant pointer cannot store the address that's stored in a constant pointer. If this were to be allowed, then the non-constant pointer could modify the constant value in the constant variable pi.

3. What is the output of the program below?

```
#include <iostream>
using namespace std;
int main() {
        int x = (9/10) * 10;
        cout << *(&x) << endl;
}
```

Solution:

0

4. What is the output of the program below?

```
#include <iostream>
using namespace std;
int main() {
        int x = 100;
        int *px = &x;

        *px++;

        //What about (*px)++;?

        cout << *px << endl;
}
```

Solution:

The output should be a random number. *px++ applies the dereferncing operator (*) on the memory address stored in px and then advance the memory address by the size of int. So px now points to a different memory address which has a random value there when dereferenced.

5. Please use pointers to implement mystrcpy() to copy the c-string pointed to by str2 to the c-string pointed to by str1. You implementation should make the program produce the following outputs:

```
C++
Pointers
Pointers
Pointers
```

Solution:

```
#include <iostream>
#include <cstring>
```

```
using namespace std;

void mystrcpy(char *str1, char*str2)
{
        while (*str1++ = *str2++);
}

int main()
{
        char str1[15] = “C++”;
        char str2[15] = “Pointers”;

        cout << str1 << endl << str2 << endl;
        mystrcpy(str1, str2); // copy from str2 to str1

        cout << str1 << endl << str2 << endl;
}
```

6. What is the output of the program below?

```
#include <iostream>
using namespace std;
int mystrlen(char *p)
{
        int len = 0;
        while (*p++ != '\0')
                len ++;
        return len;
}

int main()
{
        char str1[] = "C++";
        char str2[] = "Pointers are very powerful!";
        cout << mystrlen(str1) << endl;
        cout << mystrlen(str2) << endl;
}
```

Solution:

3
27

7. What is the output of the program below?

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

bool findValue(int *x, int n, int value)
```

```
{
        int i;
        for (i = 0; i< n; i++) {
                if (*(x+i) == value)
                        return true;
                // *(x+i) is the same as x[i]
        }
        return false;
}
int main() {
        int x[5] = {1,2,3,4,5};
        int value = 3;
        if (findValue(x, 5, value))
                cout << “Found “ << value << endl;
}
Solution:
Found 3
```