OOS PART 1 – BASIC C++

Exercise Sheet 3 – Basic Pointer Test

This lab's exercises are slightly different and emphasise that you must understand pointers to move on to the next part of the module. What follows are 13 pointers test questions which you must complete in the following manner: read a question decide on the correct response (if you are uncertain refer to the lecture notes) and only when you have made a decision enter the code fragment into a program and test the real result. In each case always decide the answer before you test the program. Exercises 14 and 15 are not pointer based, *but will test the C++ you have learned up to now* - consideration of appropriate function use and an overall design is vital!

Exercises

1.

```
int a;
int* p;

a = 2;
p = &a;
a = a + 1;
cout << *p;
```

a) 2 b) 3 c) Won't run

2.

```
int a;
int* p;

a = 2;
p = a;
a = a + 2;
cout << *p;
```

a) 2 b) 4 c) Won't run

3.

```
int a;
int b;
int* p;

p  = &a;
*p = 4;
p  = &b;
*p = 3;
cout << a << " " << b;
```

a) 4 3 b) 3 3 c) Won't run

4.

```
int a;
int b;
int* p;
int* q;

a = 3;
p = &a;
q = p;
*q = *q + 5;
cout << *p;
```

a) 8 b) 3 c) Won't run

5.

int a;
int* p;

```
a = 4;
p = &a;
cout << (*p) / a;
```

a) 1 b) 4 c) Won't run

6.

```
string s;
string* p;

s = “Fred Jones”;
p = &s;
cout << *p;
```

a) Fred Jones b) Fred c) A hexadecimal memory address

7.

```
string s;
int* i;

s = “Fred Jones”;
i = &s;
cout << *i;
```

a) Fred Jones b) A garbage number c) Won't run

8.

```
function doubleref(int* p) {(*p) = (*p) * 2;}

int main()
{
   int a = 5;
   doubleref(&a);
   cout << a;
}
```

a) 5 b) 10 c) Won't work

9.
```
int a;
int b;
int* p;
int* q;

a = 3;
p = &a;
q = p;
b = 4;
*q = b;
cout << *p << a;
```

a) 4 3 b) 3 4 c) 4 4

10.
```
int a;
int* p;

a = 3;
p = &a;
cout << p;
```

a) 3 3 b) A hexadecimal memory address c) Won’t run

11.
```
int a;
int* p;

a = 4;
p = &a;
cout << (*p+1);
```

a) 4 b) 5 c) Random Garbage

12.
```
int a;
int* q;

a = 4;
q = &a;
cout << *(q+1);
```

a) 4 b) 5 c) Random Garbage

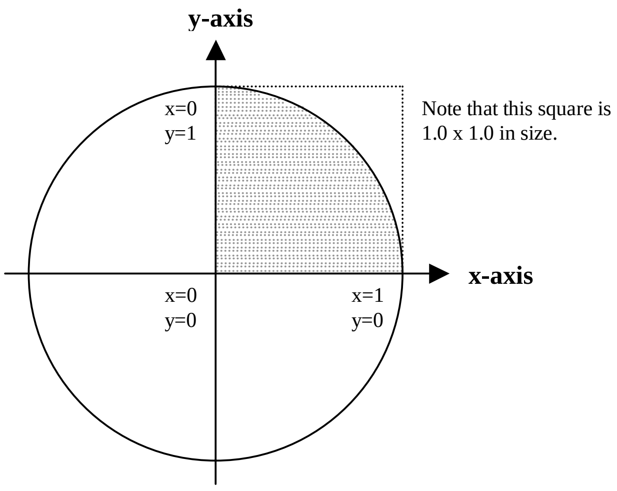
13.
```
int a;
int* p;
char b;

a = 52;
p = &a;
b = *p;
cout << b;
```

a) 52 b) A c) Won’t Run

14. THE WEEKEND TEASER:
Write a C++ program to calculate the mathematical constant of PI:

(a) Imagine a circle, whose area is given by: Area=PI*R^2
(b) Now imagine that you divide the circle exactly into 4 quadrants.
(c) The area of one quadrant is hence: Area=0.25*PI*R^2
(d) Now let us set the radius to be R=1;
(e) The equation hence becomes: Area=0.25*PI
(f) So we can simply say: PI = Area*4



Now the big question is how do you calculate the area of the grey shaded quadrant of the circle above? Use a random number generator and guess effectively the correct answer. (Hint: use rand() from <cstdlib> and make sure that you divide through by RAND_MAX.)

The technique you should use is to:
- guess a random number between 0.00 and 1.00 and assign this to x.
- guess a random number between 0.00 and 1.00 and assign this to y.
- find out if this (x,y) coordinate lies inside the grey shaded quadrant or outside. *Hint: sqrt(*x^2+y^2*).*
- Repeat these steps N times, where N is sufficiently big and then find the ratio of points inside the circle divided by the total number N, to determine the area of the grey shaded area.
- Apply this to your equation to find the mathematical constant PI.

15. Rewrite the above program slightly so that you can enter the number of decimal points precision for Pi that you want to achieve, and it will print out Pi to this precision, as well as the value of N needed to achieve this. (A suggestion – please test your program with quite small values of N).

All mistakes on lab sheet 3 were unintentional, and a consequence of me being overworked by the big cruel university ☺