CS 121 - Week 11 Worksheet Part 2 - More Pointers Practice

Questions:

1. Declare a pointer to a float and a pointer to a char.

float* fptr;
char* cptr;

2. What do we use the *sizeof()* operator for? If we have two integers, *ax* and *bx*, defined one after the other, how can we simulate the purpose of the *sizeof()* operator?

Use the *sizeof()* operator to get the byte-size of a variable or data type.

```
int ax, bx;
cout << &bx << " - " << &ax << " = "
     << ( int(&bx) - int(&ax) ) << endl;  // the space between each address is
                                          // sizeof(int)
```

Example output: http://ideone.com/gC7ThD

```
0xbfbd9f7c - 0xbfbd9f78 = 4
```

3. Explain, in words, what the following code snippet does. Make sure to use the words "dereference", "pointer" and "reference (&)" inside your explanation.

```
double num1 = 7.0;              // Define a double variable named num1, set it to 7.0
double& num2 = num1;            // Define a reference to num1 named num2
double* dptr = &num1;           // Define a double pointer named dptr and point it to
                                // the address num1 is located at.

cout << num1 << endl;           // Print out num1's value.
cout << &num1 << endl;          // Print out the address num1 is located at

num2 = 33.4;                    // Set num2 (i.e. num1) equal to 33.4
*dptr = 1234.567;               // Set the variable located to where dptr is pointing
                                // to as 1234.567.

cout << num2 << endl;           // Print out the value located at num2 (i.e. num1)
cout << &num2 << endl;          // Print out the address num2 refers to (i.e. num1)
```

4. Suppose the operating system you were working on defines a pointer to an int as 8 bytes (64 bits) wide. How wide would a pointer to a short be? A string pointer? Explain.

Both a pointer to a short or a pointer to a string would be of size 8 bytes as well as any other pointer defined on this operating system.

As an address on a 64 bit operating system is 64 bits wide (or 8 bytes, as 8 bits is 1 byte), and a pointer of any type holds an address, each pointer would thus be 8 bytes as well.

5. Explain, using your knowledge of the relationship between pointers and arrays, on why the following code produces its output.

Syntax	Output

`#include <iostream>`
`using namespace std;`
 	
`void printArrayMultByTwo(int * int_list, const int SIZE)`
`{`
`  for(int i = 0; i < SIZE; ++i)`
`  {`
`    int_list[i] *= 2;`
`    cout << int_list[i] << " ";`
`  }`
`  cout << endl << endl;`
`}`
 	
`int main()`
`{`
`  const int LIST_SIZE = 5;`
`  int my_list[] = {27, 32, 55, 1, 3752};`
 	
`  printArrayMultByTwo(my_list, LIST_SIZE);`
`  printArrayMultByTwo(my_list, LIST_SIZE);`
`  printArrayMultByTwo(my_list, LIST_SIZE);`
 	
`  return 0;`
`}`
	54 64 110 2 7504
	
	108 128 220 4 15008
	
	216 256 440 8 30016
	
	int_list is a pointer that points to the memory location (the address) of where *my_list* is located. From here, it has access to the entirety of *my_list* by using offsets (i.e. using the bracket [] operator, or pointer arithmetic).
	
	This page has some really good info on the relationship between arrays and pointers.

6. Using the code on the left, make a main function that creates a pointer that dynamically creates a Rectangle, sets its width to 3.0 and height to 4.0, prints out the rectangle, and deletes the Rectangle afterwards. Assume all the methods are well-defined.

Given Code	Your Main Function

`class Rectangle`
`{`
`    double width, height;`
`  public:`
`    Rectangle();  // defaults width = height = 1.0`
`    Rectangle(double w, double h);`
 	
`    // getters`
`    double getWidth();`
`    double getHeight();`
 	
`    // setters`
`    void setWidth(double w);`
`    void setHeight(double h);`
 	
`    // miscellaneous`
`    double getArea();`
	`int main()`
	`{`
	`  Rectangle* my_rect = new Rectangle(3.0, 4.0);`
	`  // or`
	`  //   Rectangle* my_rect;`
	`  //   *(my_rect).setWidth(3.0);`
	`  //   *(my_rect).setHeight(4.0);`
	
	`  *(my_rect).printSelf();`
	`  // or:`
	`  //   my_rect->printSelf();`
	
	`  delete my_rect;`
	`  return 0;`

`    void printSelf();` `};`	`}`