[*Practice Problems for Pointers*]

Solutions located at the end of this document

1. What will the following code print?

```
#include<iostream>

int main( ){

    int *a = new int;
    *a = -37;
    int *b;
    b = a;
    *b += 10;
    std::cout<<*a <<std::endl;

    delete a;
    delete b;
    return 0;
}
```

2. What will the following code print?

```
#include<iostream>

int main( ){

    double* x;
    double num = 24.7;
    x = &num;
    num += 5;
    std::cout<<*x <<std::endl;

    return 0;
}
```

3. Will this create a memory leak? Explain why or why not. If yes, explain how to fix it.

```
#include<iostream>

int main( ){

    double* pX = new double;
    *pX = 54;
    pX = new double;
    *pX = 89;

    delete pX;
    return 0;
}
```

******************Solutions on next page******************

Solutions

1. This will print -27. First we create a pointer called **a** and ask the computer for enough memory to hold one integer. Then we store the number -37 in that memory location and create another pointer called **b**. When we reach the line that says "**b = a**" this tells **b** to point to the same memory location that **a** is pointing to. Finally we increase the integer that **b** is pointing to by 10 and print the integer that **a** is pointing to. Since **a** and **b** are pointing to the same memory location, the value -27 will be printed.

2. This will print 29.7. First we create a pointer of type double called **x** and we create a variable of type double called **num** and store the number 24.7 in that memory location. Then we store the address of **num** in **x**, this means that **x** is now pointing to the memory location of where the double **num** is stored. When we increase **num** by 5 and print ***x**, the program will print 29.7 because **x** is pointing to **num**. Notice how we do not have to delete the pointer in this situation since we do not ask for memory using the keyword **new**, the computer will automatically free the memory that is being used by regular variables.

3. Yes, this will create a memory leak. In the first step we create a pointer of type double called **pX** and ask the computer to give us enough memory to hold one double. The computer gives us enough memory and stores the address of that memory in **pX**. In the next line we store the number 54 in that memory location. In the following line we ask the

computer for another memory location to hold another double using the same pointer. The computer gives us enough memory to hold a double and stores that memory location in **pX**. Now we have lost the address of the double that stored the number 54 which means we cannot free this memory and the computer will not be able to use that memory until you restart the computer. Then we store the number 89 in **pX** and delete **pX**, but when we delete **pX** we are only freeing the memory that is holding the 89, not the 54. If you want to fix the memory leak, you would have to delete **pX** before you ask for a new memory location.

The following code fixes the memory leak.

```
#include<iostream>

int main( ){
    double* pX = new double;
    *pX = 54;
    delete pX;  //free memory before asking for a new
                //memory location
    pX = new double;
    *pX = 89;
    delete pX;
    return 0;
}
```