AERSP 424: Advanced Computer Programming

Submission Instructions: Homework 3
Due: 1/29/19

Submission Instructions:

- Submit the file with the .cpp extension containing your C++ source code.
- Notes on submissions:
  - Submit single cpp file for questions 1,2
  - Submit one pdf for questions 3 and 4. NO HANDWRITTEN SUBMISSIONS.
  - Complete submissions should include a cpp file and a pdf.
- Teams of up to 3 are allowed. Put the names of all team members on a single submission.

1. Implement the following statements:

   a. Create a double variable called **dFirst** and initialize **dFirst** to 1.1.

   ```
   double dFirst=1.1;
   ```

   b. Create a pointer of type **double** called **dPtr1** .

   ```
   double* dPtr1;
   ```

   c. Assign the address of **dFirst** to **dPtr1.**

   ```
   dPtr1 = &dFirst;
   ```

   d. Print the value of the object pointed to by **dPtr1**. Print the address of **dPtr1**. Print the address of object pointed to by **dPtr1.**

   ```
   cout << "value of dPtr1: "<<*dPtr1 <<endl;
   cout << "address of dPtr1: "<<&dPtr1 <<endl;
   cout << "address of object pointed to by dPtr1: "<<dPtr1 <<endl;
   cout << "Sanity check: "<<&dFirst <<endl;
   ```

   e. Create a **double** variable called **dSecond** and initialize **dSecond** to 3.14.

   ```
   double dSecond =3.14;
   ```

   f. Create a pointer of type **double** called **dPtr2** . Assign the address of **dSecond** to **dPtr2.**

```
double* dPtr2;
dPtr2 = &dSecond;
```

g. Print the value of the object pointed to by **dPtr2**. Print the address of **dPtr2**. Print the address of object pointed to by **dPtr2.**

```
cout << "value of dPtr2: "<<*dPtr2 <<endl;
cout << "address of dPtr2: "<<&dPtr2 <<endl;
cout << "address of object pointed to by dPtr2: "<<dPtr2 <<endl;
```

h. Set **dPtr1** to **dPtr2.**

```
dPtr1 = dPtr2;
```

i. Print the value of the object pointed to by **dPtr1**. Print the address of **dPtr1**. Print the address of object pointed to by **dPtr1.**

```
cout << "value of dPtr1: "<<*dPtr1 <<endl;
cout << "address of dPtr1: "<<&dPtr1 <<endl;
cout << "address of object pointed to by dPtr1: "<<dPtr1 <<endl;
```

j. Did the assignment of **dPtr1** to **dPtr2** cause a memory leak? Explain why or why not.

No. The pointers do not point to variables created using the new command. The memory from these variables will be deallocated when the function returns.

2. Implement the following statements:

a. Use the **new** keyword to create a pointer to a **float** called **fPointerToFloat**. Initialize the value of this variable to 6.43.

```
float* fPointerToFloat = new float(6.43);
```

b. Print the value of **fPointerToFloat,** the pointer's address, and the address pointed to by **fPointerToFloat**.

```
cout << "Value pointed to: " << *fPointerToFloat << endl;
cout << "Pointer's address: "<< &fPointerToFloat << endl;
cout << "Address pointed to: "<< fPointerToFloat << endl;
```

c. Use the **`new`** keyword to create a pointer to a **`float`** called **`fPointerToFloatTwo`**. Initialize the value of this variable to 3.14.

```
float* fPointerToFloatTwo = new float(3.14);
```

d. Print the value of **`fPointerToFloatTwo,`** the pointer's address, and the address pointed to by **`fPointerToFloatTwo`**.

```
cout << "Value pointed to: " << *fPointerToFloatTwo << endl;
cout << "Pointer's address: "<< &fPointerToFloatTwo << endl;
cout << "Address pointed to: "<< fPointerToFloatTwo << endl;
```

e. Set **`fPointerToFloatTwo`** to point to the same address as **`fPointerToFloat.`**

```
fPointerToFloatTwo = fPointerToFloat;
```

f. Print the value that **`fPointerToFloatTwo`** points to. Do the same for **`fPointerToFloat.`**

```
cout << "Value pointed TWO: " << *fPointerToFloatTwo << endl;
cout << "Value pointed: "<< *fPointerToFloat << endl;
```

g. Explain what happened to the memory location which contained 3.14. Can we access it? If so, how?

`The memory location that contained 3.14 still exists but we cannot access it.`

h. Delete the memory pointed to by the pointers and set the variables to zero.

```
delete fPointerToFloatTwo;
delete fPointerToFloat;
fPointerToFloatTwo = 0;
fPointerToFloat = 0;
```

3. What is the output of the following? **Add comments briefly explaining each line.**

a.

```
#include <iostream>
using namespace std;
int main()
{
    int num[5];
    int* p;
    p = num;
    *p = 10;
    p++;
    *p = 20;
    p = &num[2];
    *p = 30;
    p = num + 3;
    *p = 40;
    p = num;
    *(p + 4) = 50;
    for (int i = 0; i < 5; i++)
        cout << num[i] << ", ";
    return 0;
}
```

```
     10, 20, 30, 40, 50,
```

b.

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = { 4, 5, 6, 7 };
    int* p = (arr + 1);
    cout << *arr + 10<<endl;
    return 0;
}
```

```
14
```

c.

```
#include <iostream>
using namespace std;
int main()
{
    int a = 32, *ptr = &a;
    char ch = 'A', &cho = ch;


    cho += a;
    *ptr += ch;
    cout << a << ", " << ch << endl;
    return 0;
}
```

```
129, a
```

4. Identify and correct all errors in each of the following statements. Potential errors include compile errors, link errors, and logical errors.

```
char *cPtr;
char *cPtrPtr=0;
char array[] = {"I love pointers"};
char cCharacter='a';
```

a.

```
//set cPtr pointer to cCharacter
cPtr = cCharacter;

should be:
cPtr = &cCharacter;
```

b.

```
//increment the value cPtr to the next character
cPtr = cPtr + 1;

should be:
*cPtr =*cPtr+ 1;
```

c.

```
//set cPtr to the letter 'v' in the array
cPtr = array[ 2 ];

should be:
cPtr = &array[4];
```

d.

```
//use cPtr to print out the "love pointers" portion of
//the array
for( int i=0; i<=7; i++)
     cout << cPtr[ i ] << endl;

should be:
cPtr = &array[0];
for( int i=0; i<6;i++)
       cout << *(cPtr+i) << endl;
```

e.

```
// set the double pointer cPtrPtr to cPtr
cPtrPtr = cPtr;

should be:
cPtrPtr = &cPtr;
```

f. **// print out the value pointed to by cPtrPtr**
   cout << *cPtrPtr << endl;

should be:

```
cout << *cPtrPtr << endl;
```