CSI 32 Extra credit assignment – part 3

Answers and solutions

Topics:

- dynamic memory allocation,
- references
- pointers

1. (2 pts) Give an example of a memory leak (as a code fragment):

```
int* a = new int; // allocated memory for integer value, a is a pointer to that memory
a = new int; // allocated another memory slot for an integer value, a got its address;
// memory leak – we no longer have access to the first allocated memory slot
```

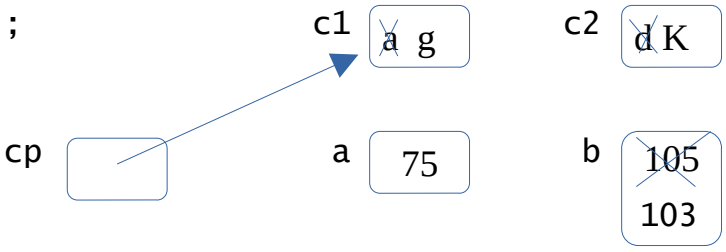
2. (2 pts) Give pictorial representation of the memory for the following code fragment:
(show all the work, from the very first line of code till the very last one, do it as one picture)

```
char c1 = 'a', c2 = 'd';

int a = 75, b = 105;

char* cp = &c1;

*cp = 'g';
c2 = a;
b = c1;
```



Comments: ASCII code of letter K is 75 `(c2 = a` replaces `d` with `K)`
ASCII code of g is 103 `(b = c1` replaces `105` with `103)`

3. (3 pts) Define three functions with name that find the sum of squares of three floating point values, that are supplied as parameters, and returns the result by value;

- The first function takes those parameters by value, and is called `sumOfSquares()`
- the second function takes those parameters by reference, but their modification is prohibited, and is called `sumOfSquaresRef()`
- the third function takes those parameters by pointers, but their modification is prohibited, and is called `sumOfSquaresP()`

Here is an example of the first function call:

```
float x = 9.8, y = 1.2, z{2.3}, result;
result = sumOfSquares(x,y,z);
```

Answer:

```
// The first function takes those parameters by value
double sumOfSquares(double a, double b, double c)
{
	return a + b + c;
}

// the second function takes those parameters by reference,
// but their modification is prohibited, and is called
double sumOfSquaresRef(const double& a, const double& b, const double& c)
{
	return a + b + c;
}

// the third function takes those parameters by pointers,
// but their modification is prohibited, and is called
double sumOfSquaresP(const double* a, const double* b, const double* c)
{
	return *a + *b + *c;
}
```

You can follow up with some testing.