ME/ECE/EMA/CS 759
High Performance Computing for Engineering Applications
Assignment 1
Date Assigned: September 9, 2013
Date Due: September 16, 2013 [11:59 PM]

1. Summarize a list of three good coding practices that you believe in. In one or two sentences comment on each of them. If necessary, use code snippets to make your point across.
You might find these references useful:
- http://en.wikibooks.org/wiki/C%2B%2B_Programming/Code_Style
- http://en.wikipedia.org/wiki/Hungarian_notation

Answer:

The three good coding practices are:
1. **Indentation**: Indent styles assist in identifying the control flow and blocks of code. Indentation and white space do not affect function, although logical and consistent indentation makes code more readable.

2. **Vertical Alignment:** It is always helpful to align similar elements vertically so that it becomes easier to debug the typo mistakes in the program code.

3. **Tabs:** The use of tabs to create white space presents particular issues when not enough care is taken because the location of the tabulation point can be different depending on the tools being used and even the preferences of the user.

As an example, one programmer prefers tab stops of two and has his toolset configured this way, and uses these to format his code.

```
int     ix;     // Index to scan array
long    sum;    // Accumulator for sum
```

Another programmer prefers tab stops of four, and his toolset is configured this way. When he examines his code, he may well find it difficult to read.

```
Int             ix; // Index to scan array
Long    sum; // Accumulator for sum
```

2. The purpose of this exercise is to understand how to generate an executable on Euler, the cluster that we'll be using in ME759. To this end, write a very short program whose output is "Hello! I'm student XYZ.", where XYZ is the set of three most significant digits of your student ID. For instance, if I was student 903422, the output would be "Hello! I'm student 903." You might want to use the g++ or gcc compiler to compile/link. To get the first three digits, use the itoa function or friends to convert your id to a string, and then pick up the relevant three characters of the string.

Solution Code:

```
# include<stdlib.h>
#include<stdio.h>

int main()
{
int i;
long int id = 903422;
char buff[10];

sprintf(buff,"%ld",id);
printf("Hello! I'm a student");
for(i-0;i<3;++i)
printf("%c",buff[i]);
return 0;
}
```

3. Write a C program that reads a string that is provided as a command line argument. Pass the string to a function that you write. Within the function count the total number of characters in the string excluding the null character. Return this value and print it out in the main program. We are going to test your program by passing it all sorts of strings: from empty strings, to strings that are 256 characters. We won't pass it a string that is longer than that.

Solution:

```
#include<stdio.h>
void main(int argc, char* argv[])
{
int ctr;
for(ctr=1;ctr<argc;ctr++)
{

printf("\n command line argument %d=%s",ctr,argv[ctr]);
}
}
```

4. The purpose of this problem is to learn how to use the gdb debugger under Euler and to understand better how pointer arithmetic works. To this end, you will have to use the flag –g when compiling your code with g++ to include debug information in the executable. Consider the code in the text-box below. Use the gdb debugger to step through the code and answer the following questions:

```
#include<iostream>

int main() {
int d;
char c;
short s;
int* p;
int arr[2];

p = &d;
*p = 10;
c = (char)1;

p = arr;
*(p+1) = 5;
p[0] = d;

*( (char*)p + 1 ) = c;

return 0;
}
```

a) What is the value stored in p at various times in the program, and why? What is the size of this variable on Euler?

Solution:
At line #11 value stored in p= 0x7fffffffdcd0
At line #15 value stored in p= 0x7fffffffdcc0

The size of(p) = 8

b) What is the address of p and c?
Solution:
Address of p = 0x7fffffffdcd8,
Address of c = 0x7fffffffdcd5

c) What is the value of arr[0] after the assignment on line 16?
Solution:
At line # 16: arr[0]= 0.

d) What is the value of arr[0] at the end of the program?
Solution:
At the end of the program arr[0] = 266
.
e) Explain: (*i*) why the value of arr[0] changes; and (*ii*) why exactly you got the value that you got.

Solution:

0000	0000	0000	0000	0000	0001	0000	1010

4-byte (arr[0])

At line # 10: arr[0]= 0

At line # 18: arr[0]= 10: The value of arr[0] changes at *((char*)p +1)=c because the pointer p of integer type points to array arr, which is equal to p and after p[0] = d, the value of arr[0] changes to 10.

At line # 20: arr[0]= 266: The value of arr[0] changes after *((char*)p +1)=c because the pointer points to the 2^{nd} byte which has the value 1 and so the value when converted to decimal becomes (2^8 + 2^3 + 2^2) = 266