University of California, San Diego
Dept. of Computer Science and Engineering
CSE 30 – Computer Organization and Systems Programming
Problem Set #3

Problem 1: Logical Operations

Write the function `isolateByte` that takes as arguments two integers, and returns the byte at the `byte_number`. For example, in following function, if `number = 0x12345678` and `byte_number = 2`, then it should return the `0x34`. You should only use bitwise and logical operations. You can assume that `byte_number` is between 0 and 3 and `number` is 32 bits.

```
char isolateByte(const int number, unsigned int byte_number)
```

Problem 2: Multiplication without '*'

a) Write the function `multiplyByPowerOf2` that performs multiplication by any power of 2 without using * and / operators or any C library functions. For example, `multiplyByPowerOf2(7, 3)` returns 7 * 2^3 = 56.

```
int multiplyByPowerOf2(int n, unsigned int power)
```

b) Can we implement a function that multiplies a non-"power of two" number (e.g, 3,5,6,7,9,10,...) without using the * operators? If it is possible, give a non-"power of two" example. Or explain why it is not possible.

```
int multiplyByN(int number, unsigned int n)
```

Problem 3: String Functions

Write a function `cse30strncat` functions which appends the first `num` characters of `source` to `destination`, plus a terminating null-character. If the length of the C string in `source` is less than *num*, only the content up to the terminating null-character is copied.

`destination` - Pointer to the destination array, which should contain a C string, and be large enough to contain the concatenated resulting string, including the additional null-character.

`source` - C string to be appended.

`num` - Maximum number of characters to be appended.

```
char * cse30strncat(char * destination, char * source,  unsigned int num)
```

You cannot use any C library functions (e.g., from <string.h>).

Problem 4: String List Functions

a) Using the code given in the lecture (http://cseweb.ucsd.edu/~kastner/cse30/lecture1.25.c), write the function `deleteNodeAfter` that deletes a given node from the list. For instance, assume `NodeOne`, `NodeTwo` and `NodeThree` are valid nodes and the following code is executed:

```
addNodeAfter(NodeOne, NodeTwo)
addNodeAfter(NodeTwo, NodeThree)
```

Then executing `deteteNodeAfter(NodeOne)` will result in `NodeOne` pointing to `NodeThree` and all of the memory associated with `NodeTwo` freed.

```
void deleteNodeAfter(struct StringListNode * node)
```

b) Write a function `deleteAllNodes` that deletes all nodes, including the node passed in as an argument as well as all of the nodes in the list after this node.

```
void deleteAllNodes(struct StringListNode * node)
```

Problem 5: 2D Matrices

Write the function `copy2DArray` that copies contents of one 2D array to a new 2D array. Assume that the array is square (i.e., `size` x `size`). You must use pointers to reference elements (e.g., `*array`) and not array indexing (e.g., `array[3]`). You should return a pointer to the first element of the new array.

```
int * copy2DArray(int * array, int size)
```

Problem 6: I Palindrome I

Write a function `palindrome` that checks if the string is a palindrome and returns true or false. A palindrome is a sequence of characters that is the same when read backwards and forwards. You cannot use C library functions (e.g., `<string.h>`).

```
int palindrome(char *s)
```

Problem 7: ARM Arithmetic

Translate the following code into ARM assembly:

```
Z = (A+B) * (C+D) * (E+F)
```

Assume that variables `A-F, Z` are in registers `r0-r6`, respectively (e.g. `C = r2`). Also, assume that you cannot overwrite variables `A-F` since they will all be used later in the program.

a) Assume that you have sequential processor where `ADD` and `MUL` take 1 and 3 cycles, respectively. Write the code such that it uses the minimum number of registers and cycles. How many additional registers (other than `r0-r6`) does your code require? How many cycles does your code need?

b) Assume that you have a different type of processor architecture (call it VLIW) that can perform one `ADD` operation and one `MUL` operation during every cycle. Rewrite the code to take advantage of this and use the minimum number of cycles. The VLIW processor uses a faster implementation of `MUL`, which only takes 1 cycle, the same as the cycle time of `ADD`. To make it easier to grade, please write `MUL` and `ADD` that are executed in the same cycle on one line, for example:

```
MUL r1, r2, r3              ADD r4, r2, r8
```

How many cycles does your code need?