Homework 2: C Programming
Friday, January 15, 2010
Due: Thursday, January 21, 2010

Problem P1 [10, 10 points]: C as a Programming Language
Explain the reasons why C is considered a “low level” programming language?
Sometimes, C is also called a “system programming language”. Why?

Problem P2 [15 points]: Coding in C
The following code counts the number of 1s in the binary representation of an integer. It
has bugs. Fix them by changing no more than three lines. Do not delete or add any lines.

```c
void printOnes (int x)
{
    int i;
    int numOnes = 0;
    for (0 = i; i < 32; i++) {
        if ((x & 0x1) = 1)
            numOnes++;
        x = x >> 1;
    }
    printf("The number of ones is %d\n", numOnes);
}
```

Problem P3 [10 points]: Coding in C
Write a function named `printBin()` that prints the binary representation of an unsigned
integer number. For this problem you should assume that an integer is only 32 bits,
meaning it has values between 0 - 4294967295. There is no need to remove leading 0s
(i.e., for 6 000000110 is just as acceptable an answer as 110). Please compile and test
your program to ensure that it works correctly.

```c
#include <stdio.h>
int main ( )
{
    printf("# binary representation of 0, should be: 0 it is: "); printBin(0);
    printf("# binary representation of 234567890, should be:
           11011111011001110011010010 it is: "); printBin(234567890u);
    printf("# binary representation of 523746375, should be:
           11111001101111011101000110011 it is: "); printBin(523746375u);
    printf("# binary representation of 667, should be: 1010011011 it is: ");
    printBin(667u);
    return 0; }
void printBin(unsigned int n) {
    /* your code here */
    printf("\n\n"); }
```
Problem P4 [10 points]: Pointers
Given the following variable declarations, which of the following are valid expressions?

- int x = 1;
- int* p = &x;

1. (p + x)
2. (p - x)
3. (p == p)
4. (x + p)
5. (x - p)

6. (x == p)
7. (p + p)
8. (p - p)
9. (p == 0)
10. (p == NULL)

Problem P5 [10 points]: Arrays
What is the output of the following C code?

```c
#include <string.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    const char *strs[] = {
        "FLKAIEL9KJLFNJSDF*%!SSJFIPESAYANIC*RN",
        "JAGWUFHWNJOGSYU",
        "GLKA%HR9K&FEKJTX",
        "8LKAIHL9KOJDFLKJII"
    };
    func(strs[1], 1);
    func(*(strs+2), 3);
    func(*strs, 5);
}

int func(char *start, int inc)
{
    char *p;
    const int len = strlen(start);
    for (p = start; p < start + len; p += inc)
        printf("%c", p);
    printf("\n");
}
```

Problem P6 [20 points]: Arrays
A stringList is a list data structure where the values are a char * (string). It is defined exactly the same as the strings list in the previous problem. Furthermore, define “head” as the pointer to the first element of the string list. A sample stringList looks like this:
Translate that following memory layout into a stringList similar to the one above. The variable “head” is 0x4000 0000. The strings are coded as ASCII characters, e.g. ‘A’ = 0x41. You can find hex to ASCII tables easily by searching the web. Note: Each string is null terminated, ending with 0x00, strings begin with the rightmost hex values. Ex: 0x00434241 = ‘ABC’ terminated with a null character.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFF FFFF</td>
<td></td>
<td>0x0010 0000</td>
<td></td>
</tr>
<tr>
<td>0x4000 0000</td>
<td>0x0001 1000</td>
<td>0x0004 1FFF</td>
<td>0x0001 0000</td>
</tr>
<tr>
<td>0x1000 0000</td>
<td>0x0004 6F00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4E4F 4954</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x415A 494E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0800 0000</td>
<td>0x4147 524F</td>
<td>0x0001 0000</td>
<td>0x0045 5343</td>
</tr>
<tr>
<td></td>
<td>0x0000 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x52455455</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0303 EA00</td>
<td>0x0100 0000</td>
<td>0x0000 3000</td>
<td>0x504D4F43</td>
</tr>
<tr>
<td></td>
<td>0x0000B000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0004 1FFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0200 1B04</td>
<td>0x0800 0000</td>
<td>0x0000 1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4024FFA4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0100 0000</td>
<td>0x00003033</td>
<td>0x0000 0000</td>
<td></td>
</tr>
</tbody>
</table>