Name: _____ Student ID:_____

CS 354 Practice Exam 1

This is a practice exam. It has not been carefully edited (e.g., there may be minor errors).
This practice exam is designed to be harder than the exam you will be given. You should be able to complete every question on this exam. If you feel comfortable with this practice exam you should perform well on the real exam.
Come to class Monday October 5th with any questions on this exam (or from the class). This day will be a review day.

Test details:

See http://pages.cs.wisc.edu/~cs354-1/exams.html

Oct 6th Tues 5:30 PM to 7:00 PM at

Van Vleck Room B102(Section 1), Room B130(Section 2)

Test topics:

- C Programming (K+R book and some random places in CS:APP book)
  - Compiling (preprocessor, compiler, assembler, linking & loading, execution)
  - C types & qualifiers
  - C operations
  - Pointers and arrays
  - Segments of memory / memory layout
  - Memory allocation
  - C Functions
  - Structs
  - Linked lists
- Data representation (Ch. 2 Sections 0-3 CS:APP)
  - Bits
  - Bytes, words, etc.
  - Integer representation (unsigned, two's complement)
  - Characters
  - Operations (add, sub, shift, multiply)
  - Overflow
- x86 assembly (Ch. 3 Sections 0-6,8,9,10 CS:APP (not section 7 or 11-13))
  - Registers
  - Operands (immediates, registers, addressing memory)
  - Data movement
  - Arithmetic instructions
  - Control flow instructions
  - C to assembly
  - Assembly to C

Name: _____ Student ID:_____

Question 1: Swap

(A) Write a swap function, that swaps the values held in the two pointers below. After this function, the two values should be swapped in memory. Hint: You need a ***maximum*** of 5 lines (you can do it in less too).

```
void swap(float *a, float *b) {




}
```

(B) Write the assembly language (x86) version of the same code with the following assumptions. Hint: You need a ***maximum*** of 8 lines (you can do it in less, too).
%eax contains the first parameter (`float *a`)
%edx contains the second parameter (`float *b`)

```
swap:
```

Name: _____ Student ID:_____

Question 2: x86 assembly

(A) Write the equivalent of the x86 instruction

```
push %edx
```

as a series of x86 instructions, without using the `push` instruction.

(B) Write the C equivalent of the following x86 assembly code. Assume %eax holds the C variable named intvar and %ecx holds the C variable named count. Hint: The C code should be less than 6 lines.

```
    testl %eax, %ecx
    je reset
    incl %ecx
    jmp continue
reset:
    movl $0, %ecx
continue:
```

Name: _____ Student ID:_____

Question 3: True/False

Write either "True" or "False" for each of the questions below. **If you answer "False" give an explanation or show how to change the statement into a true statement.**

The following code will be referenced for these questions.

```
int i_array[10];
char c_array[10];
int *ip;
```

Assume: `i_array` is held in the register `%esi` and `c_array` is in the register `%edi`
Note: `sizeof(int)` is 4 and `sizeof(char)` is 1.

(A) `i_array[5]` and `*(i_array+5)` is equivalent.

(B) `*ip = i_array` is a valid C statement.

(C) The following x86 statements will load the 5th element of **`i_array`** (e.g., `i_array[5]` in C) into `%eax`.

```
movl   $5, %ecx
movl   (%esi, %ecx, 1), %eax
```

(D) The following x86 statements will load the 5th element of **`c_array`** (e.g., `c_array[5]` in C) into `%eax`.

```
movl   $5, %ecx
movl   (%edi, %ecx, 1), %eax
```

(E) `array[11]` is a valid C statement (Hint: Trick question?).

(F) The four segments of memory provided every executable program are: Heap, Queue, Data, and Code.

Name: _____ Student ID:_____

Question 4: Operators, Types

a) Given A = 12, B = 24 and C = 10, what do the following C expressions evaluate to? Assume A, B and C are of type “short “(which is of size 16 bits).

1. A + B =
2. A * C =
3. B++ =
4. A % 5 =
5. A << 2 =
6. A | C =

b) Give any one example of where you would use the C basic data type “void”.

c) Give an example each for how the dot operator(.) and the arrow operator(->) are used when writing code involving structures?

Name: _____ Student ID:_____

Question 5: Data Representation

a) Fill the empty cells in the following table appropriately:

8 Bit Binary Integer	Value with Unsigned encoding	Value with 2's complement encoding
0 1 0 1 1 0 1 0		
	122	
1 0 1 0 1 1 1 0		
		-120

Name: _____ Student ID:_____

Question 6: x86 Assembly

a) Convert the following C code snippet with an “if-else-if” statement into the corresponding x86 assembly instructions.

```
if(x<=0)
{
       z = 0;
}else if(x <= 10000){
   z = 1;
}else{
   z = 2;
}
```

Assume: x is at %ebp+4, y is ta %ebp+8 and z is at %ebp+12

b) Write the x86 assembly code corresponding to the following C code:
Assume that the base address of the array “arr” is 0x100(hexadecimal).

```
int arr[4] = {1,2,3,4};
for(int i=0;i<4;i++){
     arr[i]--;
}
```

Write your x86 assembly code below:

Name: _____ Student ID:_____

Question 7: Pointers and Singly Linked Lists

a) What do each of the following two C functions do at a high level?

```
int usefulFunc1(char* s){
    int i = 0;
    if(s != NULL){
          while(s[i] != '\0'){
                i++;
           }
    }
    return i;
}

char* usefulFunc2(char *str){
       if(str == NULL) return NULL;
       int i = usefulFunc1(str);
       char* s = str;
       char* e = s + (i-1);
       while(s<e){
           char t = *s;
           *s = *e;
           *e = t;
            s++;
            e--;
       }
       return str;
}
```

usefulFunc1:

usefulFunc2:

b) Complete the C function below so that it adds a new node to the back of a singly linked.

Assume the same linked list node structure as shown in part(b) above.
Be sure handle any error scenarios in the code that you write.
The signature of the method is as follows:

- it takes the pointer to the head pointer of the singly linked list as the first argument.
- the second argument is the integer value for the data field of the new node
- it returns 0 on successful addition of a new to the back of the linked list
- it returns -1 on facing any error scenarios.

If the first argument to the function is an empty linked list, the function must update the head pointer after adding the new node appropriately.

Use this page and the following page to write your code:

```
int addBack(struct node** head, int val){
```