

CSE31 HW 1

This assignment checks your understanding of C using pointers and structs with review of number representation. You can fill in this document directly for your submission.

Problem 1

a. Given the 8 bit binary integers below, fill in the corresponding base 10 values according to the listed representations:

Binary	Unsigned	Signed	1's Complement	2's Complement	Biased
1001 0110					
0110 1001					
0011 1010					
1111 0000					

b. Fill T/F in the following table:

Property	Unsigned	Signed	1's Comp	2's Comp	Biased
Can represent positive numbers					
Can represent negative numbers					
Has more than one representation for 0					
Use the same addition process as unsigned					

c. What is the value in decimal of the most negative 16-bit 2's complement integer?

d. What is the value in decimal of the most positive 16-bit signed integer?

Problem 2

Write a C function named `copyStrArray` that, given an integer "count" and an array "strArray" that contains "count" strings, returns a pointer to a complete ("deep") copy of the array. (In Java terminology, this would be a "clone".) For example, the program segment

```
int main (int argc, char **argv) {
    char **ptr;
    ptr = copyStrArray (argc, argv);
    ...
}
```

would place in `ptr` a pointer to a copy of `argv`, the command-line argument structure. You may assume that there is sufficient free memory in which to build the copied structure. Make no assumptions about the size of a pointer or a char (ie 64-bits vs 32-bits machine). Include all necessary casts, and allocate only as much memory as necessary. You may use any function in the `stdio`, `stdlib`, or `string` libraries.

Problem 3

a. The following function should allocate space for a new string, copy the string from the passed argument into the new string, and convert every lower-case character in the **new** string into an upper-case character (do not modify the original string). Fill-in the blanks and the body of the `for()` loop:

```
char* upcase(char* str) {
    char* p;
    char* result;

    result = (char*)
malloc(_____);

    strcpy(_____, _____);

    for( p=result; *p!='\0'; p++ ) {
/* Fill-in 'A' = 65, 'a' = 97, 'Z' = 90 , 'z' = 122 */

    }
    return result;
}
```

b. Consider the code below. The upcase name() function should convert the i^{th} name to upper case by calling upcase by ref, which should in turn call upcase(). Complete the implementation of upcase_by_ref. You may not change any part of upcase_name.

```
void upcase_by_ref( char** n ) { /* Fill-in */

}

void upcase_name(char* names[], int i) { /* No not
touch */
    upcase_by_ref( &(amp;names[i]) );
}
```

Problem 4

a. Complete the following setName, getStudentID, and setStudentID functions:

```
#define MAX_NAME_LEN 128

typedef struct {
    char name[MAX_NAME_LEN];
    unsigned long sid;
} Student;

/* return the name of student s */
const char* getName(const Student* s) {
    return s->name;
}

/* set the name of student s */
void setName(Student* s, const char* name) {
    /* fill me in */
}
```

```
/* return the SID of student s */
unsigned long getStudentID(const Student* s) {
    /* fill me in */
}

/* set the SID of student s */
void setStudentID(Student* s, unsigned long sid) {
    /* fill me in */
}
```

b. What is the logical error in the following function?

```
Student* makeDefault(void) {
    Student s;
    setName(&s, "John");
    setStudentID(&s, 12345678);
    return &s;
}
```