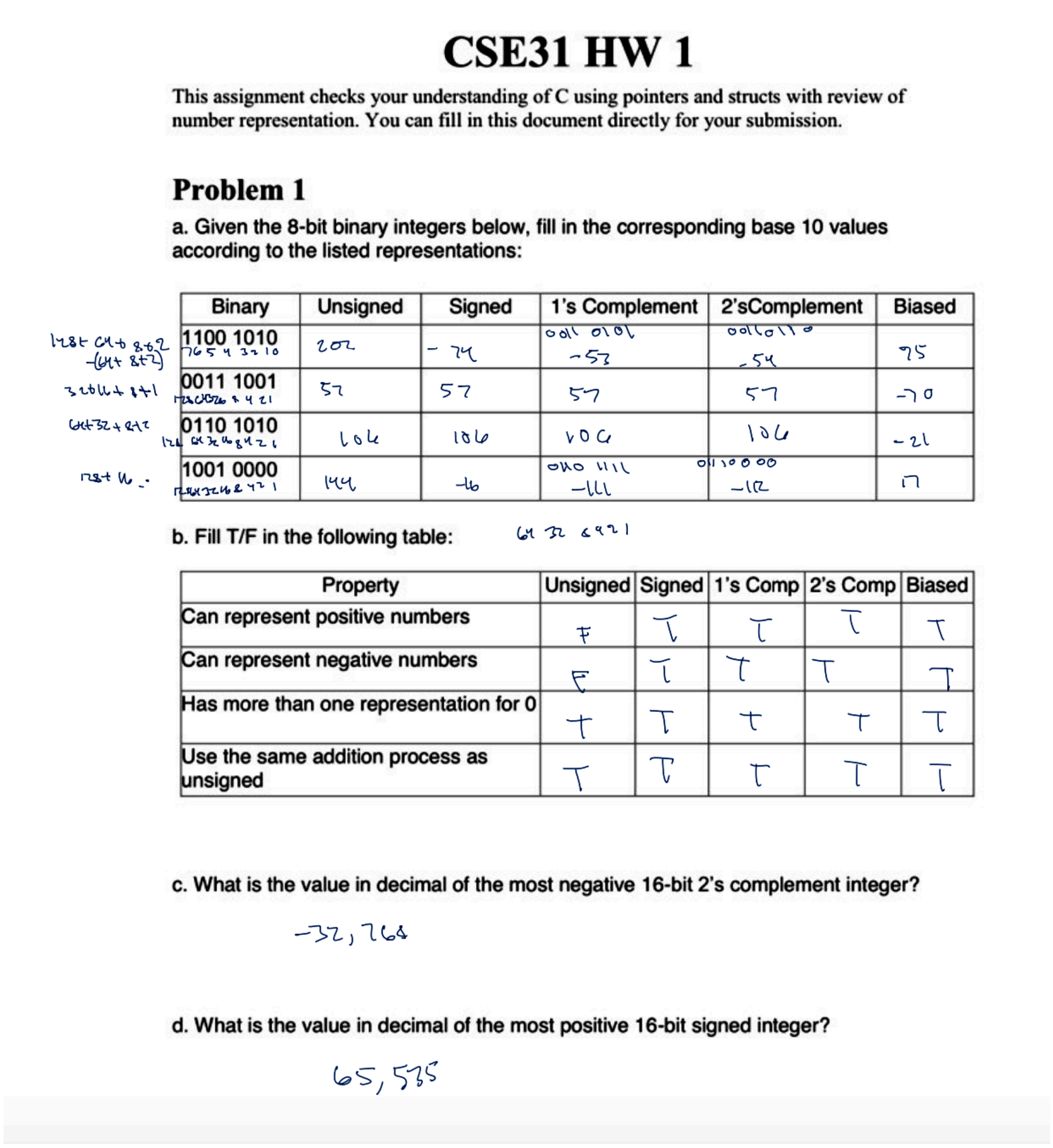
Andre Martin
02/28/2020

Done on one-note

<u>CSE31</u> HW1



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:

128+64+8+2
-(64+8+2)
32+16+8+1
64+32+8+2
128+16

Binary	Unsigned	Signed	1's Complement	2'sComplement	Biased
1100 1010 76543210	202	-74	0011 0101 -53	0011 0110 -54	75
0011 1001	57	57	57	57	-70
0110 1010	106	106	106	106	-21
1001 0000	144	-16	0110 1111 -111	0111 0000 -112	17

b. Fill T/F in the following table: 64 32 16 4 2 1

Property	Unsigned	Signed	1's Comp	2's Comp	Biased
Can represent positive numbers	F	T	T	T	T
Can represent negative numbers	F	T	T	T	T
Has more than one representation for 0	T	T	T	T	T
Use the same addition process as unsigned	T	T	T	T	T

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

-32,768

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

65,535

Andre Martin
02/28/2020

Done on one-note

Problem 2

Write a C function named **swapArray** that, given two integer arrays of size "*n*", swap the content of these arrays. For example, the program segment

```
int main (int argc, char **argv) {
  int *arr1, *arr2;
... // Assume some code here to fill-in both arrays
  swapArray(arr1, arr2, n);
... // Assume some code here to print both arrays
}
```

would print the following output if arr1 contains [10 20 30 40 50 60 70 80 90 100] and arr2 contains [0 9 8 7 6 5 4 3 2 1]:

```
arr1 after swapping: 0 9 8 7 6 5 4 3 2 1
arr2 after swapping:  10 20 30 40 50 60 70 80 90 100
```

Note: you only need to implement the **swapArray** function, no need to worry about how the main program does the input and output.

```
void swapArray( int* a1, int* a2, int size){
        int* atHold = (int*) malloc (size * sizeof(int));
        a1 = (int*) malloc ( size * sizeof (int));
        a2= (int*) malloc (size * sizeof(int));
        int i;
        for( i =0; i<size ; i++ ){
            *(atHold+i) =*(a1+i);     ← swap a1 to atHold
            *(a1+i)= *(a2+i);         ← swap a2 to a1
            *(a2+i)= *(atHold +i);    ← swap atHold to a2
        }
}
```

Problem 3

a. The following function should allocate space for a new string, copy the string from the passed

Andre Martin Done on one-note
02/28/2020

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 upper-case character in the **new** string into a lower-case character (do not modify the original string). Fill-in the blanks and the body of the ***for() loop***:

```
char* changeCase(char* str) {
     char* p;
     char* result;
     result = (char*) malloc(__Size of (Char)__________);

     strcpy(__result______, ___str_______);

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

          *p= *p-32;


     }
          return result;
}
```

b. Consider the code below. The **changeCase_name()** function should convert the ith name to lower case by calling **changeCase_by_ref**, which should in turn call **changeCase()**. Complete the implementation of **changeCase_by_ref**. You may not change any part of **changeCase_name**.

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

               *n= *n-32;

          }

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

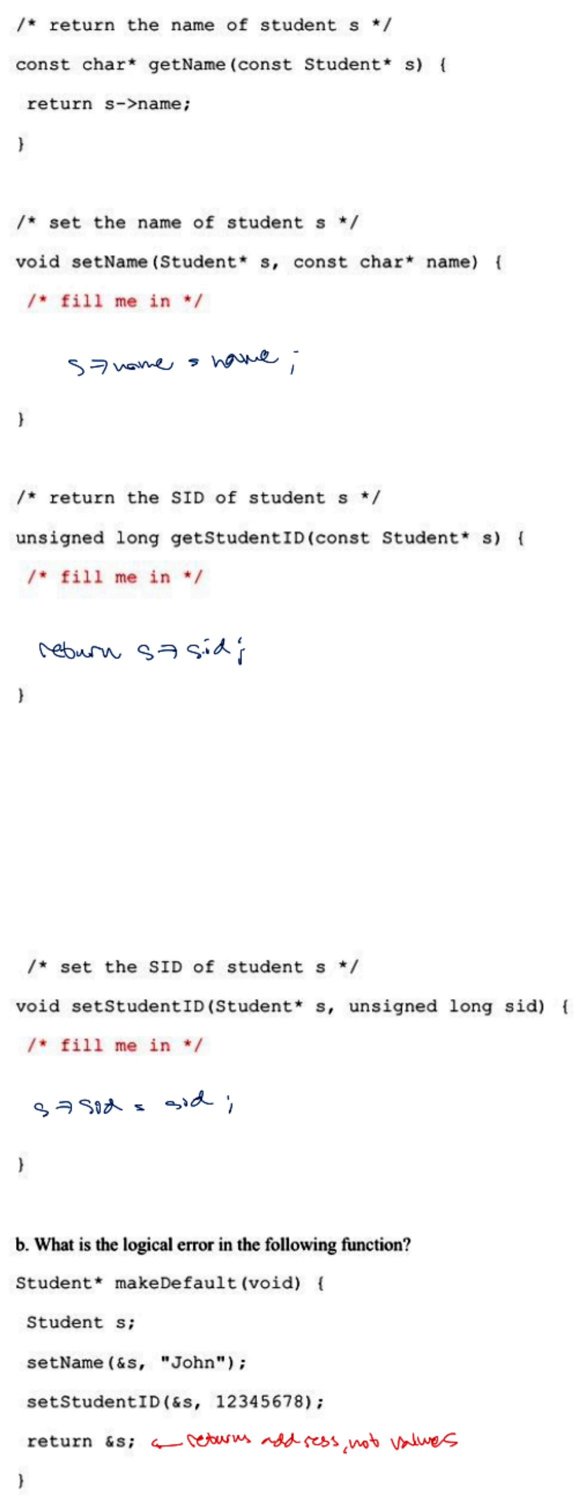
Problem 4

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;
```

Andre Martin
02/28/2020

Done on one-note



```
/* 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 */

     s->name = name;

}

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

  return s->sid;
}
```

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

 s->sid = sid;

}
```

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

```
Student* makeDefault(void) {
 Student s;
 setName(&s, "John");
 setStudentID(&s, 12345678);
 return &s; <- returns address, not values
}
```