Homework 2 - Written

Introductions

Answer these questions using the word processor of your choice. You must submit a PDF with the answers.

Part 1: Counting Operations

Each of the following C functions completes a common task using Strings. These are the kind of commands that are build into the string library and built into Python or Java.

Review each code block and answers the questions about it.

For all questions assume the number of characters in the string is n, not including the null terminator character.

Q1: Length [13 points]

```
int length(char* str){

if(str==NULL){return 0;}
int pos=0;
while(str[pos]!=0){
      pos++;
      }
return pos;
}
```

- Q1A.) How many bytes of memory are used by the function's input arguments? (1 point) 8 bytes
- Q1B.) How many bytes of memory are used by the local variables in the function? (1 point) 4 bytes
- Q1C.) How many bytes of memory are allocated on the heap in the function? (1 point) 0 bytes
- Q1D.) Complete the following table for the number of times each operations is executed in terms of n. (10 points, 2 per operation)

Operations	Count
if	1
return	1
=	1
!=	n+1
++	n

Q2: toUpper [13 points]

```
char* toUpper(char* str){
int size = length(str)+1;                # size =  n+1
char* newVer = malloc(size*sizeof(char));
for(int i=0; i < size; i++){
        if(str[i] >= 'a' && str[i] <= 'z'){
            newVer[i] = str[i] - 32;
        } else{
           newVer[i] = str[i];
    }
}
return newVer;
}
```

- Q2A.) How many bytes of memory are used by the function's input arguments? (1 point) 8 bytes
- Q2B.) How many bytes of memory are used by the local variables in the function? (1 point) 8+4+4 = 16 bytes
- Q2C.) How many bytes of memory are allocated on the heap in the function? (1 point) n+1 bytes
- Q2D.) Complete the following table for the number of times each operation is executed in terms of n. (10 points, 2 per operation)

Operations	Count
<	n+1+1 = n+2
[]	4(n+1) = 4n+4
=	3+n+1 = n+4
>=	n+1
++	n+1

Q3: replace [13 points]

```
char* replace(char* str, char oldChar, char newChar){
    int size = length(str)+1;                    # size =  n+1
    char* newVer = malloc(size*sizeof(char));
    for(int i=0; i < size; i++){
        if(str[i]==oldChar){
            newVer[i]=newChar;
        }else{
            newVer[i]=str[i];
        }
    }
    return newVer;
}
```

- Q3A.) How many bytes of memory are used by the function's input arguments? (1 point) 8+1+1 = 10 bytes
- Q3B.) How many bytes of memory are used by the local variables in the function? (1 point) 4+8+4 = 16 bytes
- Q3C.) How many bytes of memory are allocated on the heap in the function? (1 point) n+1 bytes
- Q3D.) Complete the following table for the number of times each operations is executed in terms of n. (10 points, 2 per operation)

Operations	Count
<	n+2
[]	3*(n+1) = 3n+3
=	3+n+1 = n+4
++	n+1
==	n+1

Part 2: Timing Operations

You are provided with an unfinished program written.c. Finish the code to collect the Clock Ticks (Integer) the function takes. Clock ticks are the most accurate timing mechanism available in basic C.

Remember, every execution of the code will provide slightly different clock ticks. Just pick the results of one execution. You do not need to submit your code for this question. Just the table below needs to be filled out in your PDF

Q4 - Complete the below table (11 points)

Start	Prime	Clocks
1024	1031	3
2048	2053	2
4096	4099	1
8192	8209	3
16384	16411	3
32768	32771	2
65536	65537	2
131072	131101	5
262144	262147	4
524288	524309	5
1048576	1048583	6
2097152	2097169	9
4194304	4194319	14
8388608	8388617	19
16777216	16777259	48

Start	Prime	Clocks
33554432	33554467	28
67108864	67108879	26
134217728	134217757	49
268435456	268435459	49
536870912	536870923	72
1073741824	1073741827	120