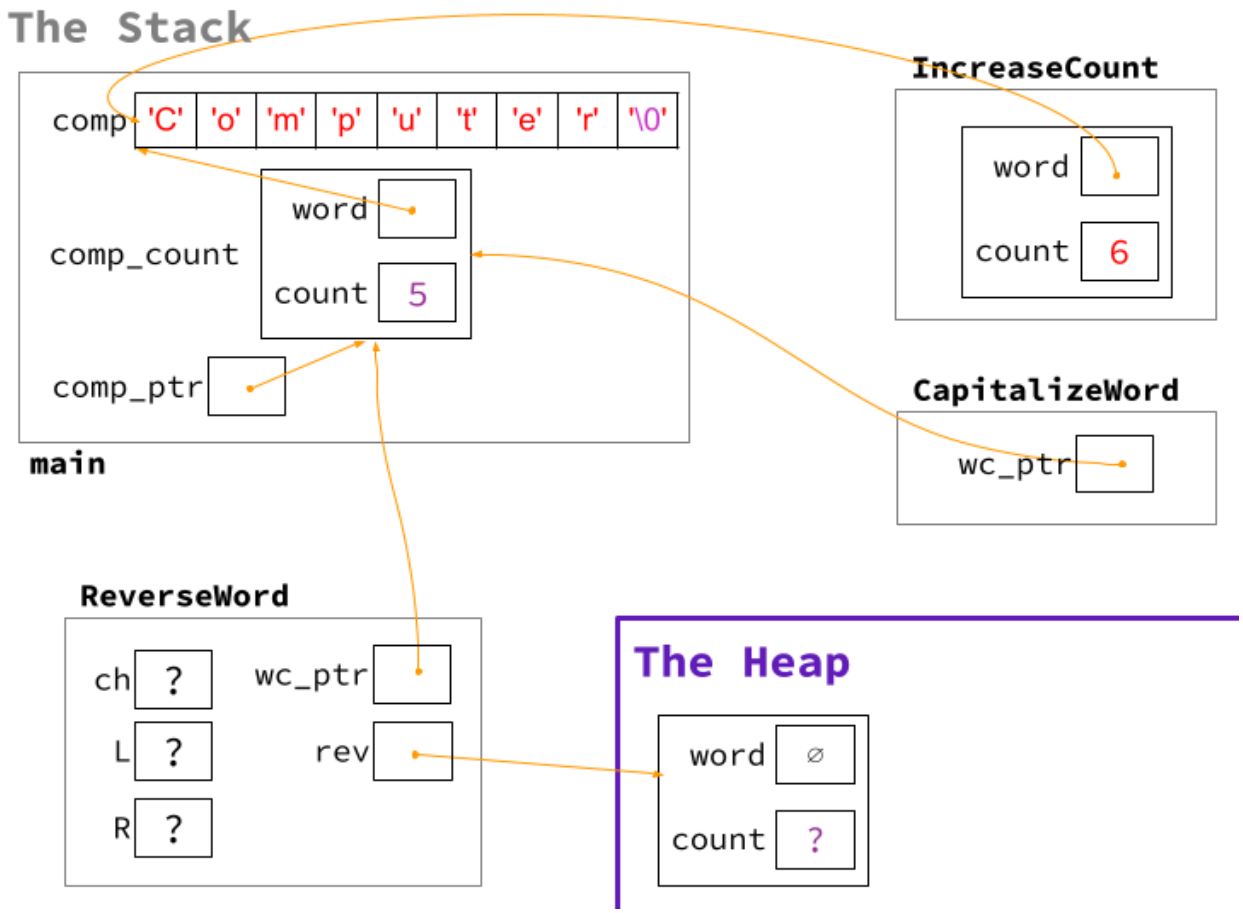
CSE 333 – Section 2: Structs and Debugging Solutions

Exercise 1

Below is the state of memory up to the call to `strcpy()` in `ReverseWord()`. Functions whose stack frames have been popped are shown for completion.



Exercise 2

Note: To see the full solution with proper use of function declarations, see https://courses.cs.washington.edu/courses/cse333/22wi/sections/02/code/wordcount_soln.c.html

```
typedef struct word_st {
  char* word;
  int   count;
} WordCount;

// Increment the count by 1
void IncreaseCount(WordCount* wc_ptr) {
  wc_ptr->count += 1;
}

// Capitalize the first letter in the word
void CapitalizeWord(WordCount* wc_ptr) {
  wc_ptr->word[0] &= ~0x20;
}

// Return a new WordCount with the letters of word in reverse
// order and a count of 0.  Returns NULL on allocation failure.
WordCount* ReverseWord(WordCount* wc_ptr) {
  WordCount* rev = (WordCount*) malloc(sizeof(WordCount));
  rev->word = (char*) malloc((strlen(wc_ptr->word) + 1)
                             * sizeof(char));
  strcpy(rev->word, wc_ptr->word);
  rev->count = 0;

  char ch;
  int L = 0, R = strlen(rev->word) - 1;
  while (L < R) {
    ch = rev->word[L];
    rev->word[L] = rev->word[R];
    rev->word[R] = ch;
    L++; R--;
  }

  return rev;
}
```

```
int main(int argc, char* argv[]) {
  char comp[] = "computer";
  WordCount comp_count = {comp, 5};
  WordCount* comp_ptr = &comp_count;

  // expecting "1. computer, 5"
  printf("1. %s, %d\n", comp_ptr->word, comp_ptr->count);
  IncreaseCount(*comp_ptr);
  // expecting "2. computer, 6"
  printf("2. %s, %d\n", comp_ptr->word, comp_ptr->count);
  CapitalizeWord(comp_ptr);
  // expecting "3. Computer, 6"
  printf("3. %s, %d\n", comp_ptr->word, comp_ptr->count);
  comp_ptr = ReverseWord(comp_ptr);
  // expecting "4. retupmoC, 0"
  printf("4. %s, %d\n", comp_ptr->word, comp_ptr->count);

  free(comp_ptr->word);
  free(comp_ptr);

  return EXIT_SUCCESS;
}
```

Exercise 3

Style Issues

1. Should error check any call to malloc()

```
if (rev == NULL) {
  return NULL;
}
if (rev->word == NULL) {
  return NULL;
}
```

2. When dealing with small structs it is appropriate to pass a copy of the struct

```
WordCount* ReverseWord(WordCount wc) { ... }
```