COP 3502 Section 1 Exam 2A – Sorting, Tries, Hash Tables, Base Conversion (Wednesday 11/15/2023)

Last Name: ___________________, First Name: ______________________

Lab Number: _________________ (Must match entry "LabNum" in Webcourses)

1) (5 pts) Show the state of the array below after each iteration of Bubble Sort. The result of the last iteration has been provided for you.

Initial Values	11	6	8	13	9	1	4
1st iteration							
2nd iteration							
3rd iteration							
4th iteration							
5th iteration							
6th iteration	1	4	6	8	9	11	13

2) (5 pts) A different implementation of the partition function (used in Quick Sort) would involve creating a separate temporary array and copying the values less than the partition element to the left side of the temporary array, and values greater than or equal to the partition element to the right side of the temporary array, and then copying everything back. A partial implementation of this function is below. Fill in the blanks to complete the implementation. (Note: For brevity, the partition element will be the item in array[low], instead of swapping a random element into that spot.)

```
int partition(int* array, int low, int high) {

    int* tmp = calloc(high-low+1, sizeof(int));
    int lowPtr = 0, highPtr = high-low;

    for (int i=low+1; i<=high; i++) {

        if (array[ ____ ] < array[ ______ ])
            tmp[lowPtr++] = array[i];
        else
            tmp[highPtr--] = array[i];
    }

    tmp[ ________ ] = array[ ______ ];

    for (int i=low; i<=high; i++)
        array[i] = tmp[i-low];
    free(tmp);

    return __________ ;
}
```

3) (10 pts) Consider the problem of determining whether or not a substring **(of unique letters)** exists in any word stored in a trie. If we solve the problem recursively, our recursive function would take in a pointer to the trie node we were "currently" on, the substring we are looking for, and an integer, k, representing the number of letters we've currently matched in the substring. (So, for example, if the substring was "rst" and k = 2, that means that the last two links we traveled down the trie to get to our current node were 'r', followed by 's'. (Thus, if the link to 't' from the current node exists, we can then return 1 to indicate that some word has the substring "rst". This is because we assume a path exists in the trie only if it eventually maps to a word.) Complete the function below to solve this task. Namely, if the substring exists amongst the words stored in the trie, the function should return 1, otherwise, it should return 0. Note: Assume that n represents the length of the substring, substr, that is being searched for, and that all characters in substr are lowercase letters. **You may assume that the string substr contains ALL UNIQUE LETTERS.** (Note: Without this assumption, this problem is much more difficult.) Fill in the blanks to complete the implementation of this function.

```
typedef struct trie {
    int isWord;
    struct trie* next[26];
} trie;

int substring(trie* root, char* substr, int k, int n) {

    if (root == NULL) return _____ ;

    if (k == n) return  _____;
    int res = 0;

    for (int i=0; i<26; i++) {
        if (i == substr[k]-'a')

            ___________________________________________________;

        else

            ___________________________________________________;
    }

   return res;
}
```

4) (9 pts) Consider implementing a hash table storing integers using the hash function shown below (value is the input to the hash function and n is the size of the hash table the integers are being stored in) and the technique of linear probing to determine collisions.

```
int hashfunction(int value, int n) {
    int res = 0;
    while (value > 0) {
        res = (res + value%10)%n;
        value = value/10;
    }
    return res;
}
```

Using a hash table of size n = 13, if we were to insert the following integers, in the order shown below, determine which indexes each number would get inserted into. (Please draw the number in the appropriate slot in the array shown below. The array is shown over two lines so that there's enough room to write each number.)

Numbers to Insert: 47312, 86297, 123456, 676, 11112335, 567, 999999, 887, and 23642.

Index	0	1	2	3	4	5	6
Value							

Index	7	8	9	10	11	12
Value						

5) (10 Pts) Perform the requested base conversions. Note: only 1 pt is given for the answers. The rest of the points are awarded for showing the proper work (using the algorithms shown in class for each conversion.)

(a) (2 pts) 89_{10} = $\_\_\_\_\_\_\_\__{2}$ (convert 89 in base 10 to base 2)

(b) (2 pts) 2310_5 = $\_\_\_\_\_\_\_\_\__{10}$ (convert 2310 in base 5 to base 10)

(c) (3 pts) 372604_8 = _______ $_{16}$ (372604 in base 8 converted to base 16)

(d) (3 pts) $FE702_{16}$ = _______________ $_8$ (FE702 in base 16 converted to base 8)

6) (1 pt) Which car company produces the popular Toyota Camry? _____________________