# CSCI-311 Assignment 4

Download Lab4.tar; to untar, use the command: tar -xvf Lab4.tar

**Task 1**. Re-write *Insertion* sort algorithm to sort suffixes of a given string S. Similarly to QuickSort, your *insertion* function will take a string S (passed constant by reference), and a vector of integers with the starting positions of suffixes of S. The function will sort the suffixes of S in the range from *low* to *high* (inclusive indices of the range). Insertion will call *lessThan* function (from Assignment 3) and will swap positions of the suffixes instead of suffixes. Here is the header of this function:

# void insertion(const string &S, vector<int> &indices, int low, int high)

### Sample input:

S="abracadabra", indices =  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ , low = 3, high = 7. insertion(S, indices, low, high); //will sort suffixes starting at positions 3, 4, 5, 6, and 7.

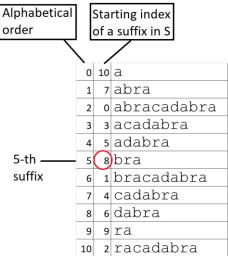
# Sample output:

After insertion is called on the *sample input*, vector *indices* will have the following order of suffix positions: indices =  $\{0, 1, 2, 7, 3, 5, 4, 6, 8, 9, 10\}$ .

**Task 2**. In this part, you need to re-write the provided code for **Selection** algorithm to work with suffixes. Given a string S, and an integer k, your program must return the starting index of the suffix that is the k-th suffix of S if the suffixes of S were sorted in alphabetical order. Remember that **Selection** algorithm does not actually sort the items in a given array.

<u>Important</u>: inside the provided <u>selection</u> code, we call STL <u>sort</u>; you need to replace this <u>sort</u> with your function <u>insertion</u>.

For example, given a string S = "abracadabra" and k = 5, your program will return 8, the starting position of the suffix "bra", that is the 5-th in the alphabetical order (the count of ordering starts with 0).



**Task 3**. Inside your main(int argc, char\* argv[]) function, do the following:

- 1. Read lines from the file whose name is given in argv[1], concatenate all lines into a single string S (just as in Assignment 3).
- 2. If argc equals to four, then create a vector of integers *indices* of size equal to the length of S, and initialize *indices*[i] = i. Then call *insertion* on S, *indices*, *low* (given in argv[2]) and *high* (given in argv[3]). After *insertion* has finished, output *indices* with a space after each element and with *endl* at the end.
- If argc equals to three, then create a vector of integers indices of size equal to the length of S, and initialize indices[i] = i. Then call selection on S, indices, low = 0, high = S.length() 1, and k (given in argv[2]). Output the returned from selection value and endl at the end.

#### Test files to test insertion and selection:

Insertion	t01, t02, t03, t04, t05, t06, t07, t08
Selection	t09, t10, t11, t12, t13, t14

Use command line arguments in the test files with extension cmd.

#### **Submission:**

Submit main.cpp to Assignment4 on turnin.

**Grading:** *Insertion* is worth 50pts and *Selection* is worth 50pts. Grading is done according how many tests will produce correct output.