**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY NOIDA**

****

**Search Engine Using Maps and Trie**

Data Structures Lab Project

B. Tech,Computer science 2nd Year (2019-2020)

Made By: Submitted To: Batch B10

NIKHIL SISODIA

18103314

**Introduction**: Search Engine Project in C++ is an academic search engine application designed to search relevant academic information and records in schools, colleges and universities. It supports two popular search engines: Google Scholar and Microsoft Academic Search. With this application, students in colleges and universities can search academic information based on the keywords and titles provided for search. The coding of this project is done in C++ language. Most academic institutions, colleges and universities, have their own websites that provide information regarding students’ research papers, project source codes and project reports. These records can be used as a reference by new students entering the colleges and universities. This search engine provides a platform to find out such records, information and data that can help students in the academic works.

**Major Data Structure Used:**

• TRIE

• Maps

Functions Used: • Insert in TRIE

void insert(struct TrieNode \*root, string key) {

struct TrieNode \*pCrawl = root;

for (int level = 0; level < key.length(); level++) {

int index = CHAR\_TO\_INDEX(key[level]);

if (!pCrawl->children[index])

pCrawl->children[index] = getNode();

pCrawl = pCrawl->children[index]; }

pCrawl->isWordEnd = true; }

• Search in TRIE

bool search(struct TrieNode \*root, const string key) {

int length = key.length();

struct TrieNode \*pCrawl = root;

for (int level = 0; level < length; level++) {

int index = CHAR\_TO\_INDEX(key[level]);

if (!pCrawl->children[index])

return false;

prefix += key[level];

pCrawl = pCrawl->children[index]; }

return (pCrawl != NULL && pCrawl->isWordEnd); }

• Auto Complete Word Suggestions

void suggestionsRec(struct TrieNode\* root, string currPrefix) {

if (root->isWordEnd) {

gotoxy(20,pos++);

cout << currPrefix;

cout << endl; }

if (isLastNode(root))

return;

for (int i = 0; i < ALPHABET\_SIZE; i++) {

if (root->children[i]) {

currPrefix.push\_back(97 + i);

suggestionsRec(root->children[i], currPrefix); }

}

}

int printAutoSuggestions(TrieNode\* root, const string query) {

struct TrieNode\* pCrawl = root; int level; int n = query.length();

for (level = 0; level < n; level++) {

int index = CHAR\_TO\_INDEX(query[level]);

if (!pCrawl->children[index]){

prefix="";

return 0;

}

pCrawl = pCrawl->children[index]; }

bool isWord = (pCrawl->isWordEnd == true);

bool isLast = isLastNode(pCrawl);

if (isWord && isLast) {

gotoxy(20,12);

cout << query << endl;

prefix="";

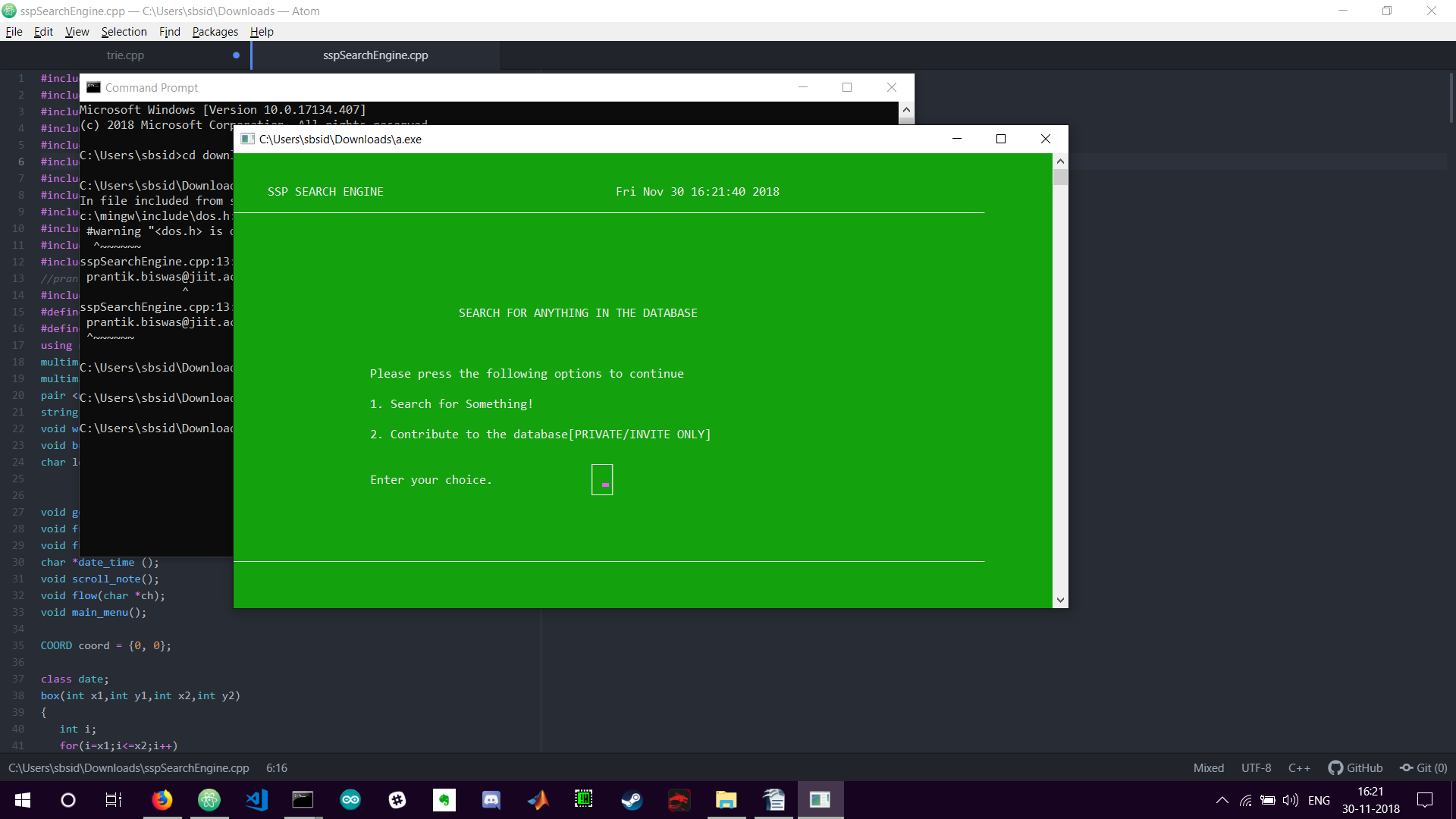
return -1; }

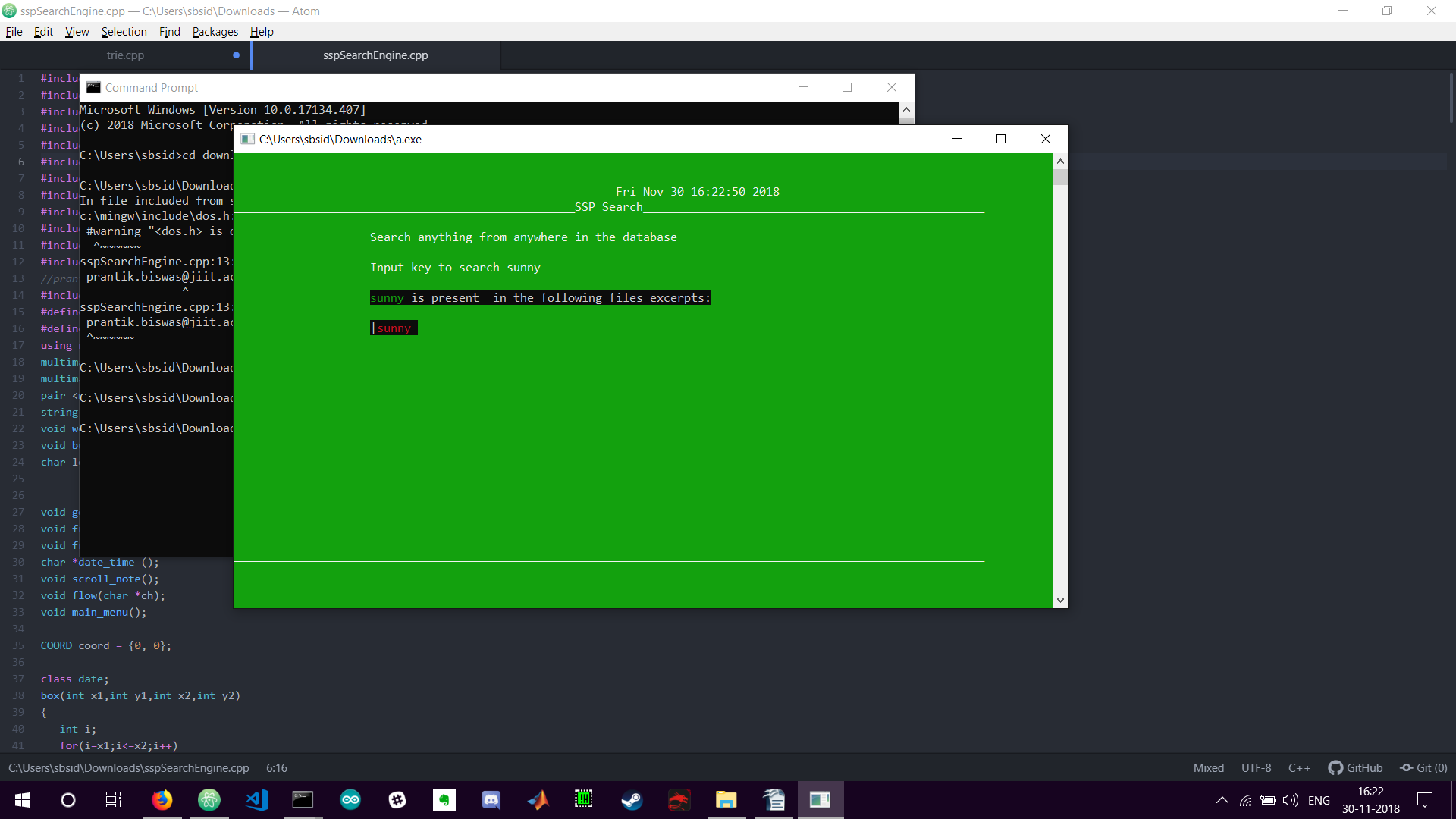
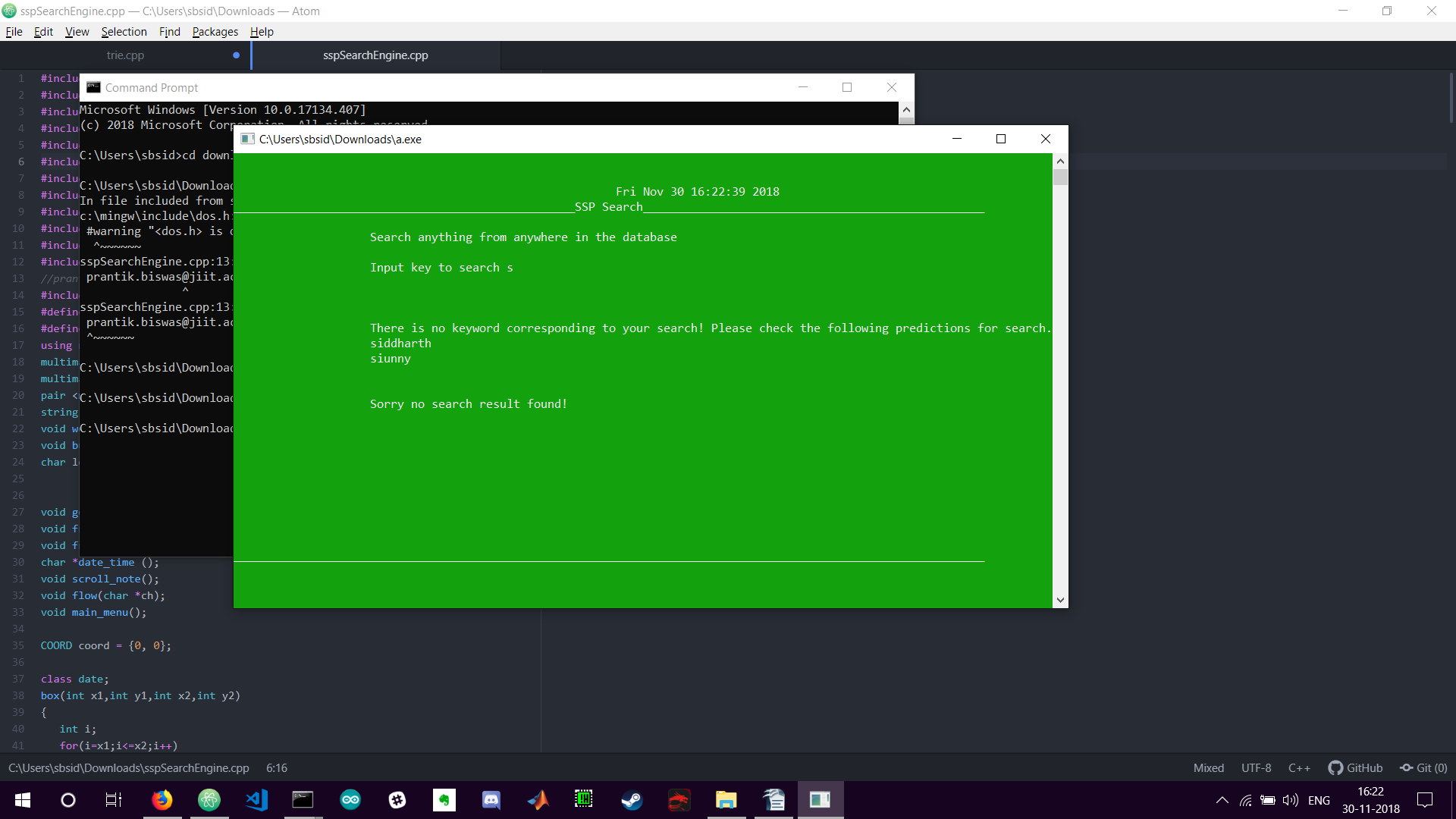
if (!isLast) { string prefix1 = query; suggestionsRec(pCrawl, prefix1);

prefix=""; return 1; }

}

**ScreenShots:**

****

****