struct node\* insert(struct node\* node, int key)

{

if (node == NULL) return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

return node;

}

struct node\* search(struct node\* root, int key)

{

if (root == NULL || root->key == key)

return root;

if (root->key < key)

return search(root->right, key);

return search(root->left, key);

}

int getKthLargestnum(int arr[], int k, int n){

priority\_queue<int> q;

for(int i=0; i<n; i++){

q.push(arr[i]);

}

for(int i=0; i<k-1; i++){

q.pop();

}

return q.top();

}

int insert(int table[], int value,int n){

int index= value%n;

int i=1;

while(table[index]!=value){

if(i>=n){

cout<<"full table";

}

index=(index+i\*i)%n;

i++;

}

table[index]=value;

}

#include <iostream>

bool search(node\* curr, int data){

if(curr-> value < data){

if(curr->right!= NULL){

search(curr->right, int data)

}

else{

node\* temp= new node()

temp->value= data;

curr->right = temp;

}

}

else if(curr->value > data){

if(curr->left!= NULL){

search(curr->left, int data)

}

}

else{

}

}

#include <iostream>

#include <string>

#include <iomanip>

#include <queue>

using namespace std;

bool isPrime(int num) {

if (num <= 1)

return false;

for (int i = 2; i < num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

int main() {

queue<int> q;

vector<int> v = { 1,2,3,4,5,6,7,8,9,10 };

for (int i = 0; i < v.size(); i++) {

if (isPrime(v[i] + 2)) {

q.push(v[i]);

}

}

while (!q.empty()) {

cout << q.front() << " ";

q.pop();

}

}

#include <iostream>

#include <string>

#include <iomanip>

#include <queue>

#include <stack>

using namespace std;

int main() {

queue<int> q;

q.push(1);

q.push(2);

q.push(3);

q.push(4);

q.push(5);

stack<int> s;

while (!q.empty()) {

s.push(q.front());

q.pop();

}

while (!s.empty()) {

cout << s.top() << " ";

s.pop();

}

while (!s.empty()) {

q.push(s.top());

s.pop();

}

}