Data Structures Interview Questions

1) What is data structure?
Data structures refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of data, but rather different set of data and how they can relate to one another in an organized manner.

2) Differentiate file structure from storage structure.
Basically, the key difference is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

3) When is a binary search best applied?
A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is search starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

5) How do you reference all the elements in a one-dimension array?
To do this, an indexed loop is used, such that the counter runs from 0 to the array size minus one. In this manner, we are able to reference all the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures applied?
Data structures is important in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?
LIFO is short for Last In First Out, and refers to how data is accessed, stored and retrieved. Using this scheme, data that was stored last, should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

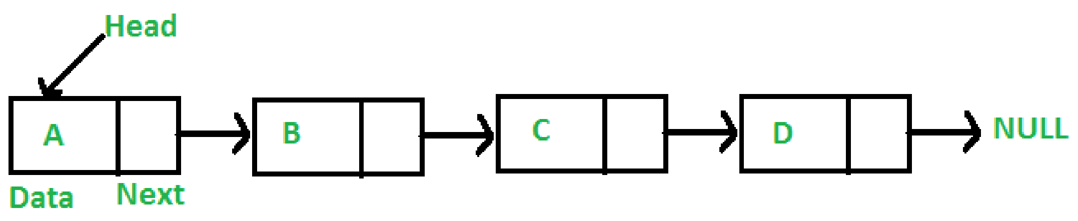
4) What is a linked list?
A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link of data storage.

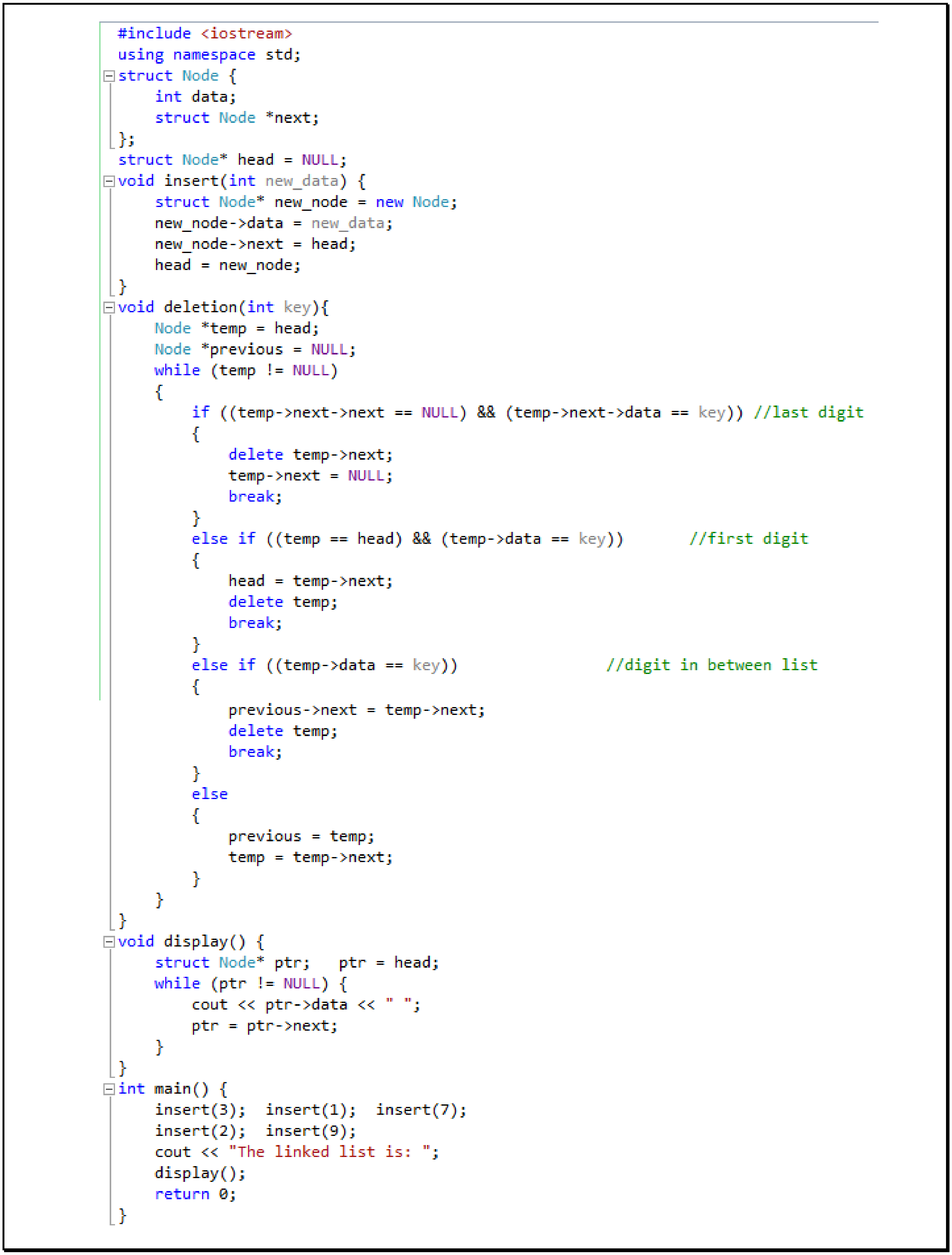
Types:

1). Singly Linked List.

2). Doubly Linked List.

3). Circular Linked List.

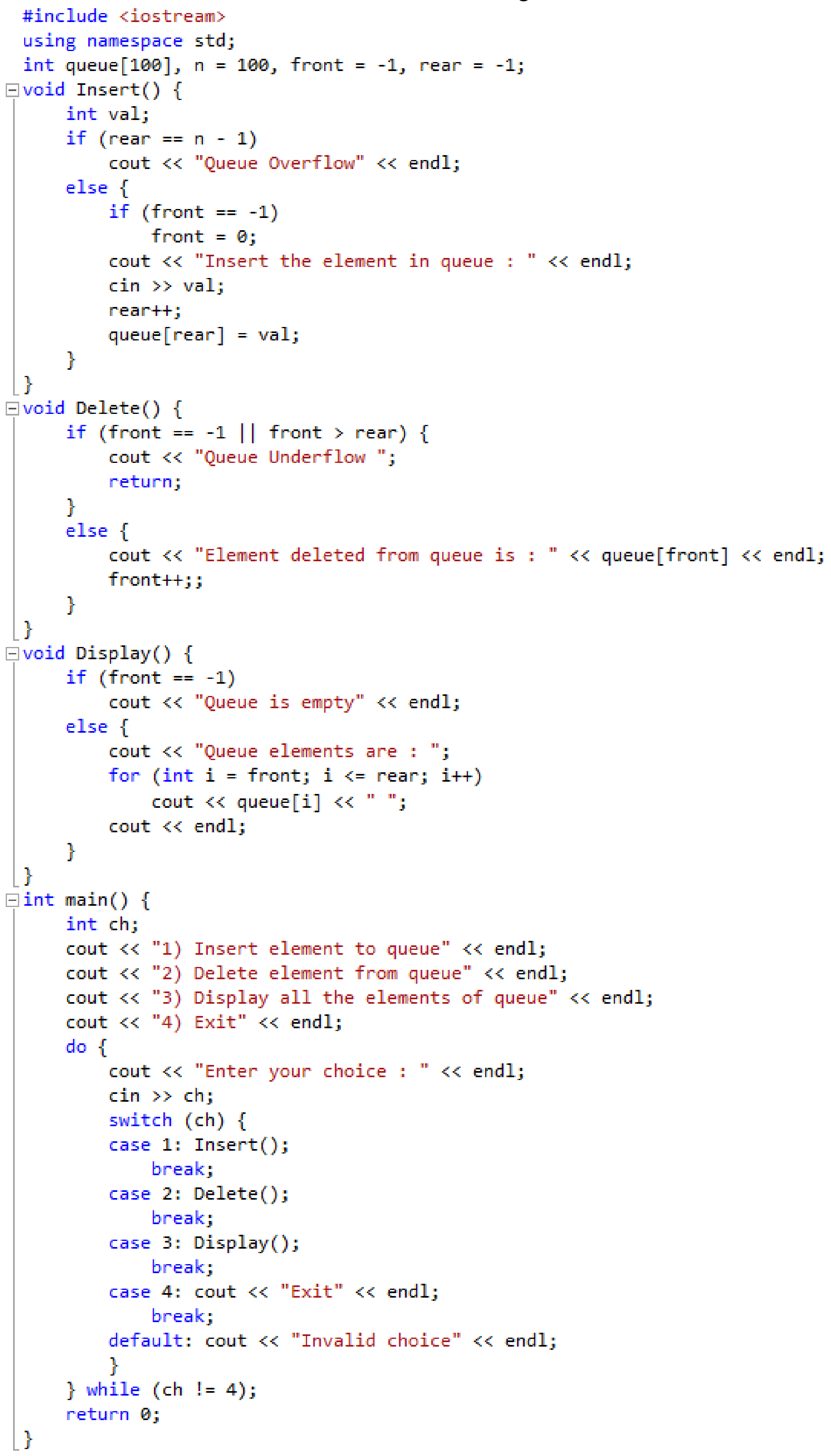




```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    struct Node *next;
};
struct Node* head = NULL;
void insert(int new_data) {
    struct Node* new_node = new Node;
    new_node->data = new_data;
    new_node->next = head;
    head = new_node;
}
void deletion(int key){
    Node *temp = head;
    Node *previous = NULL;
    while (temp != NULL)
    {
        if ((temp->next->next == NULL) && (temp->next->data == key)) //last digit
        {
            delete temp->next;
            temp->next = NULL;
            break;
        }
        else if ((temp == head) && (temp->data == key))       //first digit
        {
            head = temp->next;
            delete temp;
            break;
        }
        else if ((temp->data == key))                  //digit in between list
        {
            previous->next = temp->next;
            delete temp;
            break;
        }
        else
        {
            previous = temp;
            temp = temp->next;
        }
    }
}
void display() {
    struct Node* ptr;   ptr = head;
    while (ptr != NULL) {
        cout << ptr->data << " ";
        ptr = ptr->next;
    }
}
int main() {
    insert(3);  insert(1);  insert(7);
    insert(2);  insert(9);
    cout << "The linked list is: ";
    display();
    return 0;
}
```

8) What is a queue?

A queue is a data structures that can simulates a list or stream of data. In this structure, new elements are inserted at one end and existing elements are removed from the other end.



```
#include <iostream>
using namespace std;
int queue[100], n = 100, front = -1, rear = -1;
void Insert() {
    int val;
    if (rear == n - 1)
        cout << "Queue Overflow" << endl;
    else {
        if (front == -1)
            front = 0;
        cout << "Insert the element in queue : " << endl;
        cin >> val;
        rear++;
        queue[rear] = val;
    }
}
void Delete() {
    if (front == -1 || front > rear) {
        cout << "Queue Underflow ";
        return;
    }
    else {
        cout << "Element deleted from queue is : " << queue[front] << endl;
        front++;;
    }
}
void Display() {
    if (front == -1)
        cout << "Queue is empty" << endl;
    else {
        cout << "Queue elements are : ";
        for (int i = front; i <= rear; i++)
            cout << queue[i] << " ";
        cout << endl;
    }
}
int main() {
    int ch;
    cout << "1) Insert element to queue" << endl;
    cout << "2) Delete element from queue" << endl;
    cout << "3) Display all the elements of queue" << endl;
    cout << "4) Exit" << endl;
    do {
        cout << "Enter your choice : " << endl;
        cin >> ch;
        switch (ch) {
        case 1: Insert();
            break;
        case 2: Delete();
            break;
        case 3: Display();
            break;
        case 4: cout << "Exit" << endl;
            break;
        default: cout << "Invalid choice" << endl;
        }
    } while (ch != 4);
    return 0;
}
```

10) Which data structures is applied when dealing with a recursive function?
Recursion, which is basically a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

13) What are multidimensional arrays?
Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using a single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?
It actually depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?
Aside from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?
FIFO is short for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?
An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

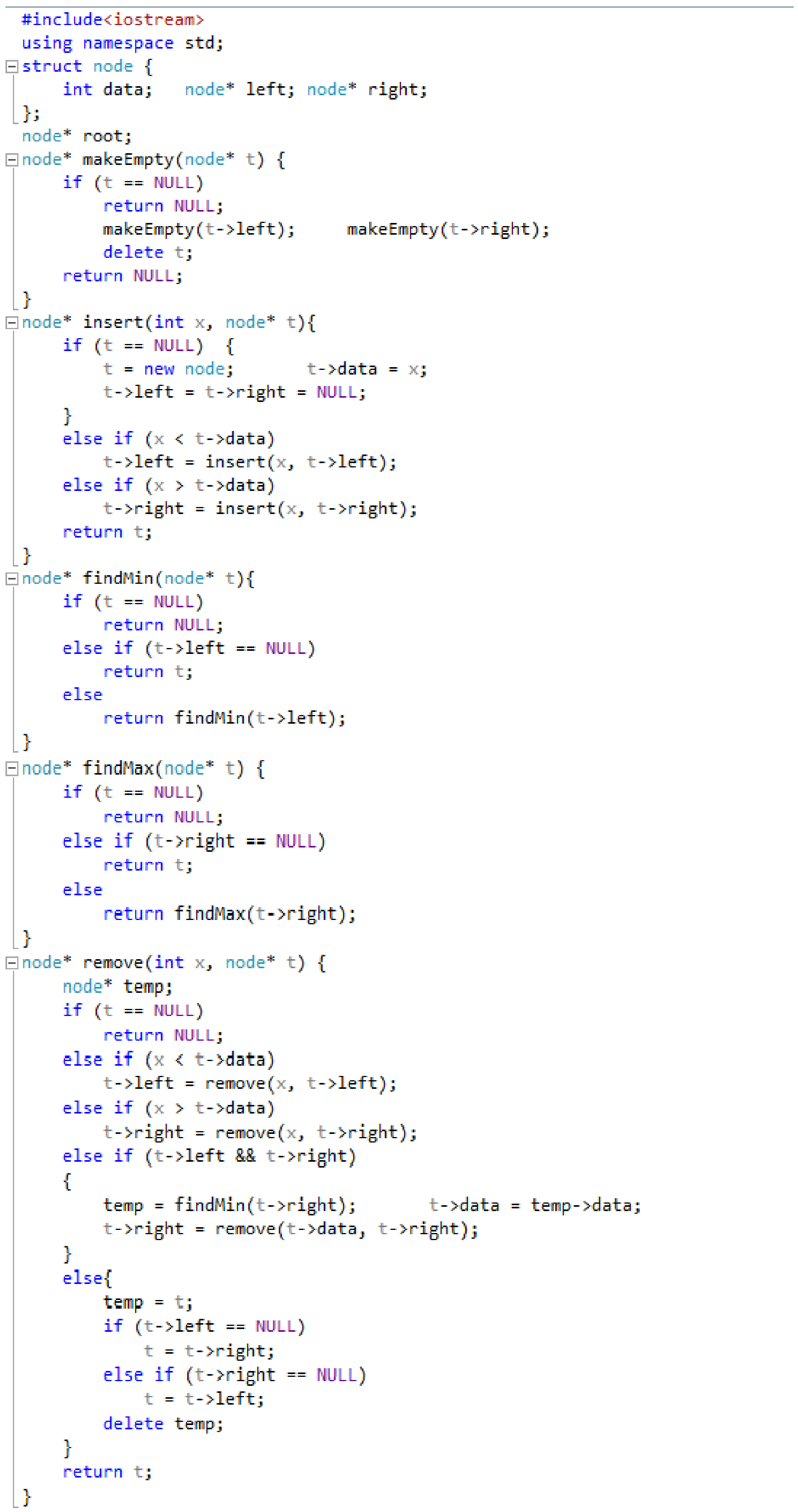
18) What is merge sort?
Merge sort takes a divide-and-conquer approach to sorting data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continuous until you have one single sorted list.

19) Differentiate NULL and VOID.
Null is actually a value, whereas Void is a data type identifier. A variable that is given a Null value simply indicates an empty value. Void is used to identify pointers as having no initial size.

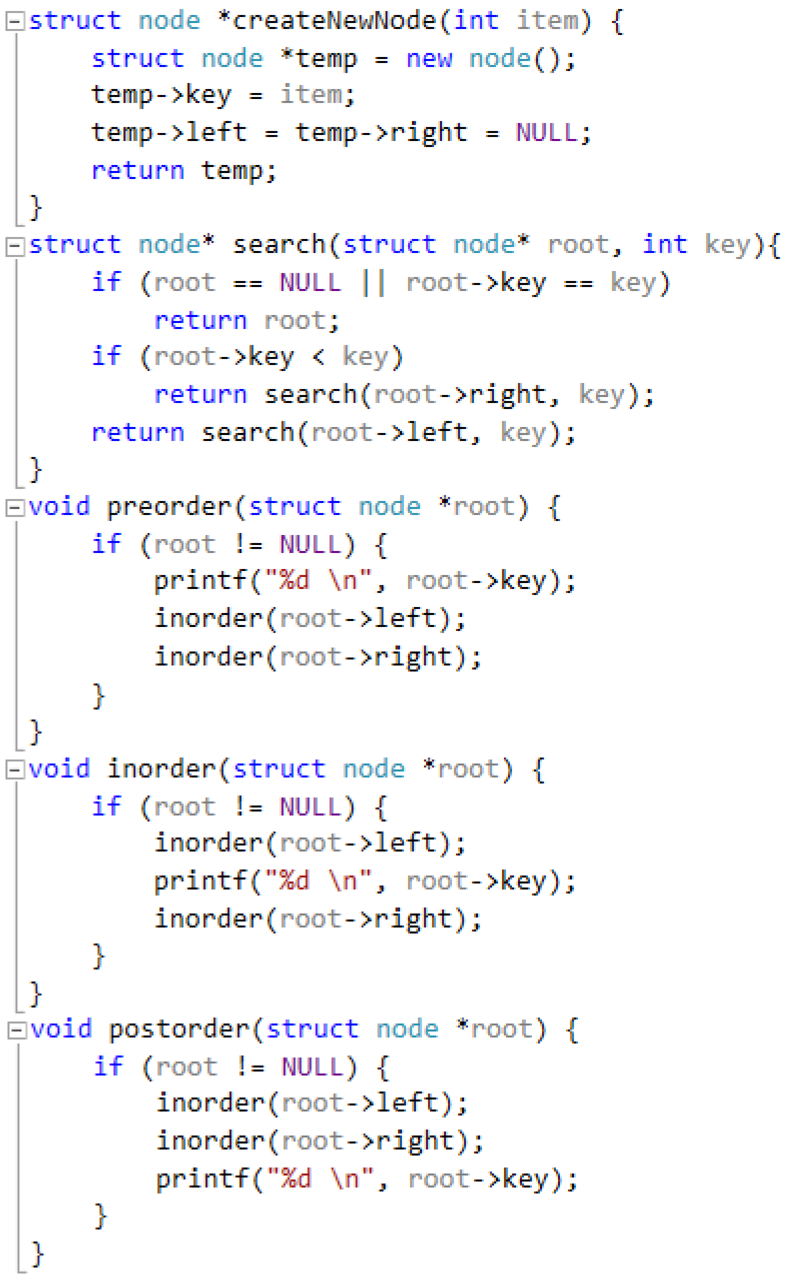
20) What is the primary advantage of a linked list?
A linked list is a very ideal data structure because it can be modified easily. This means that modifying a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?
Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular refers to the topmost data being accessed.

12) Explain Binary Search Tree
A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.



```
#include<iostream>
using namespace std;
struct node {
    int data;   node* left; node* right;
};
node* root;
node* makeEmpty(node* t) {
    if (t == NULL)
        return NULL;
        makeEmpty(t->left);     makeEmpty(t->right);
        delete t;
    return NULL;
}
node* insert(int x, node* t){
    if (t == NULL)  {
        t = new node;       t->data = x;
        t->left = t->right = NULL;
    }
    else if (x < t->data)
        t->left = insert(x, t->left);
    else if (x > t->data)
        t->right = insert(x, t->right);
    return t;
}
node* findMin(node* t){
    if (t == NULL)
        return NULL;
    else if (t->left == NULL)
        return t;
    else
        return findMin(t->left);
}
node* findMax(node* t) {
    if (t == NULL)
        return NULL;
    else if (t->right == NULL)
        return t;
    else
        return findMax(t->right);
}
node* remove(int x, node* t) {
    node* temp;
    if (t == NULL)
        return NULL;
    else if (x < t->data)
        t->left = remove(x, t->left);
    else if (x > t->data)
        t->right = remove(x, t->right);
    else if (t->left && t->right)
    {
        temp = findMin(t->right);       t->data = temp->data;
        t->right = remove(t->data, t->right);
    }
    else{
        temp = t;
        if (t->left == NULL)
            t = t->right;
        else if (t->right == NULL)
            t = t->left;
        delete temp;
    }
    return t;
}
```



```
struct node *createNewNode(int item) {
    struct node *temp = new node();
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
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);
}
void preorder(struct node *root) {
    if (root != NULL) {
        printf("%d \n", root->key);
        inorder(root->left);
        inorder(root->right);
    }
}
void inorder(struct node *root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d \n", root->key);
        inorder(root->right);
    }
}
void postorder(struct node *root) {
    if (root != NULL) {
        inorder(root->left);
        inorder(root->right);
        printf("%d \n", root->key);
    }
}
```

9) What are binary trees?
A binary tree is one type of data structure that has two nodes, a left node and a right node. In programming, binary trees are actually an extension of the linked list structures.

22) What is a linear search?
A linear search refers to the way a target key is being searched in a sequential data structure. Using this method, each element in the list is checked and compared against the target key, and is repeated until found or if the end of the list has been reached.

23) How does variable declaration affect memory allocation?
The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?
Basically, the heap is more flexible than the stack. That's because memory space for the heap can be

dynamically allocated and de-allocated as needed. However, memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?
A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?
Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?
Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?
The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. To perform this, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position. The smallest element remaining in the subarray is then located next with subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

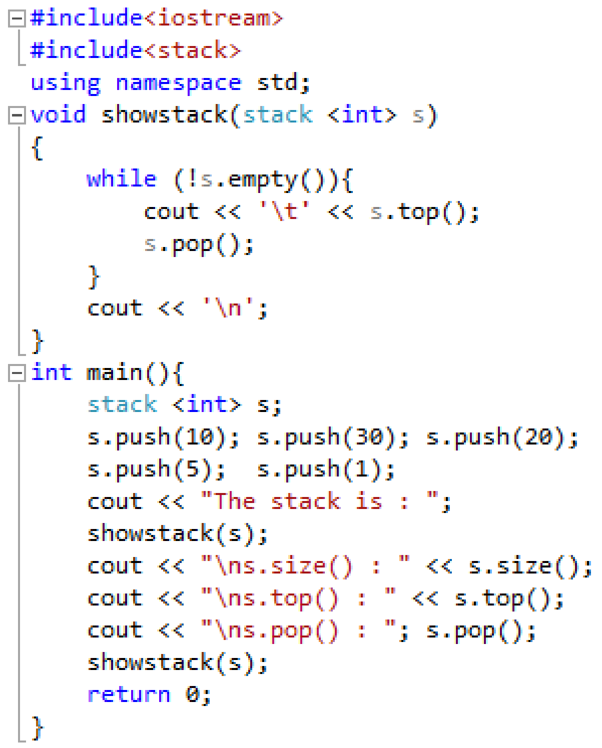
29) How do signed and unsigned numbers affect memory?
In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (unsigned 8-bit number has a range 0-255, while 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?
A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

32) In what data structures are pointers applied?
Pointers that are used in linked list have various applications in data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?
Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

11) What is a stack?
A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.



```
#include<iostream>
#include<stack>
using namespace std;
void showstack(stack <int> s)
{
    while (!s.empty()){
        cout << '\t' << s.top();
        s.pop();
    }
    cout << '\n';
}
int main(){
    stack <int> s;
    s.push(10); s.push(30); s.push(20);
    s.push(5);  s.push(1);
    cout << "The stack is : ";
    showstack(s);
    cout << "\ns.size() : " << s.size();
    cout << "\ns.top() : " << s.top();
    cout << "\ns.pop() : "; s.pop();
    showstack(s);
    return 0;
}
```

31) What are dynamic data structures?
Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

34) What are ARRAYs?
When dealing with arrays, data is stored and retrieved using an index that actually refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?
The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is intended for actual storage of data.

36) Which sorting algorithm is considered the fastest?
There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.
Data that is stored in a stack follows a LIFO pattern. This means that data access follows a sequence wherein the last data to be stored will the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.
1. if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if target is not in the root item, check if target is smaller than the root's value
5. if target is smaller than the root's value, search the left subtree
6. else, search the right subtree

39) What is a dequeue?
A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?
A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values “bubble” to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?
A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes, with each node being linked in a sequential manner.

42) How does selection sort work?
Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?
A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs, and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from non-linear data structure.
Linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks and queues. On the other hand, non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of non-linear data structure include trees and graphs.

45) What is an AVL tree?
An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?
Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and other one that links to the previous node.

47) What is Huffman’s algorithm?
Huffman’s algorithm is associated in creating extended binary trees that has minimum weighted path lengths from the given weights. It makes use of a table that contains frequency of occurrence for each data element.

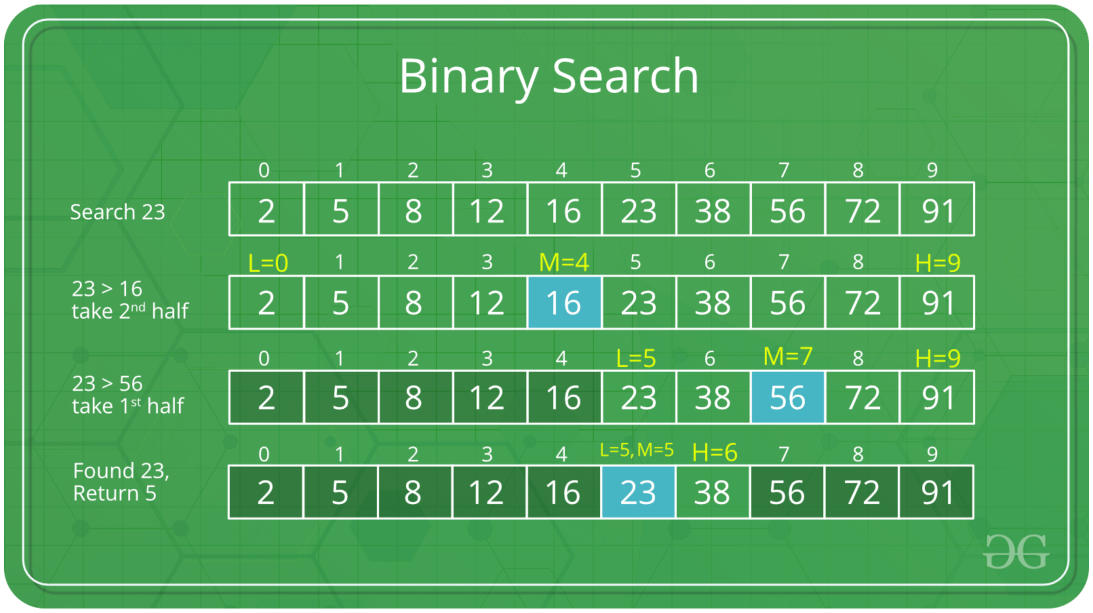
48) What is Fibonacci search?
Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can greatly reduce the time needed in order to reach the target element.

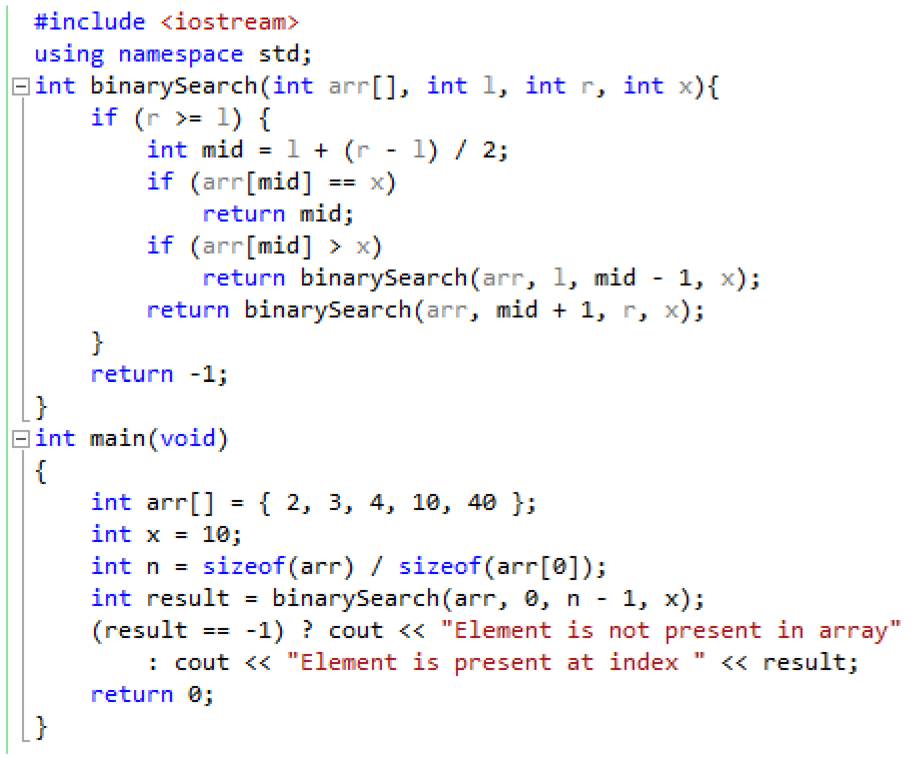
49) Briefly explain recursive algorithm.
Recursive algorithm targets a problem by dividing it into smaller, manageable subproblems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?
To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.

50) What is binary search algorithm?

A simple approach is to do search. The time complexity of above algorithm is O(n). Another approach to perform the same task is using Binary Search. **Binary Search:** Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

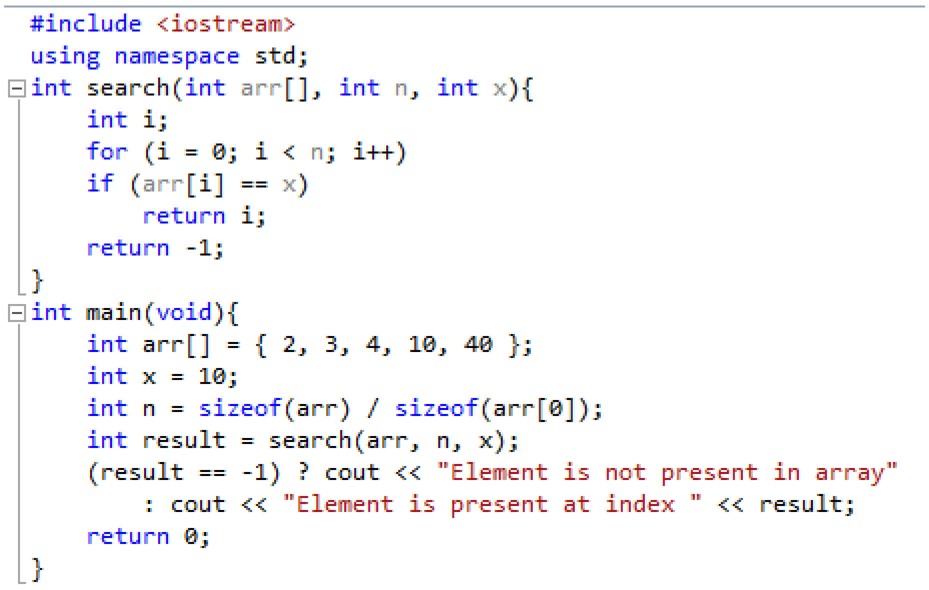




```
#include <iostream>
using namespace std;
int binarySearch(int arr[], int l, int r, int x){
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
    return -1;
}
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? cout << "Element is not present in array"
        : cout << "Element is present at index " << result;
    return 0;
}
```

50) What is Linear search algorithm?

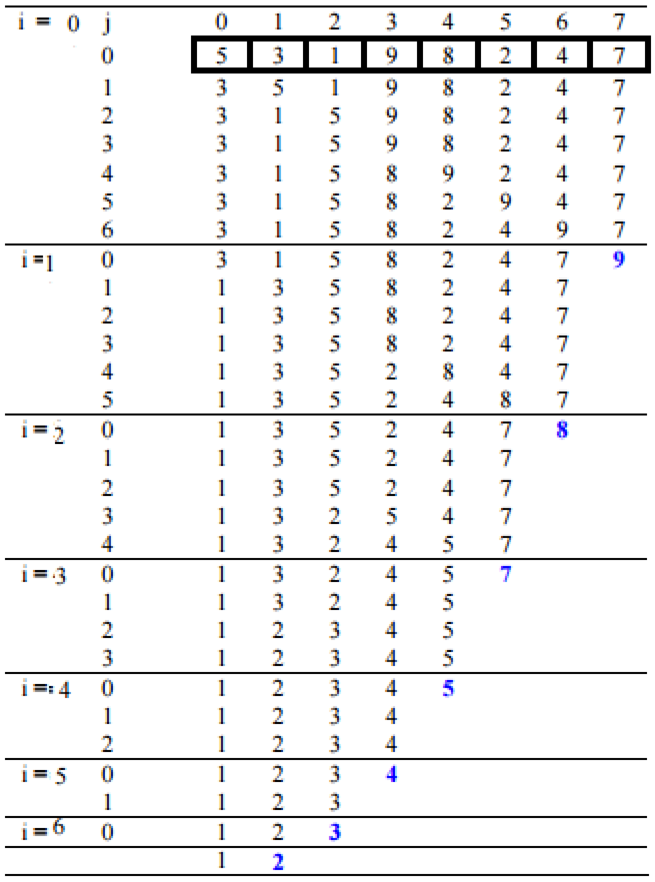
A simple approach is to do **linear search**, i.e. Start from the leftmost element of arr[] and one by one compare x with each element of arr[]. If x matches with an element, return the index. If x doesn’t match with any of elements, return -1.



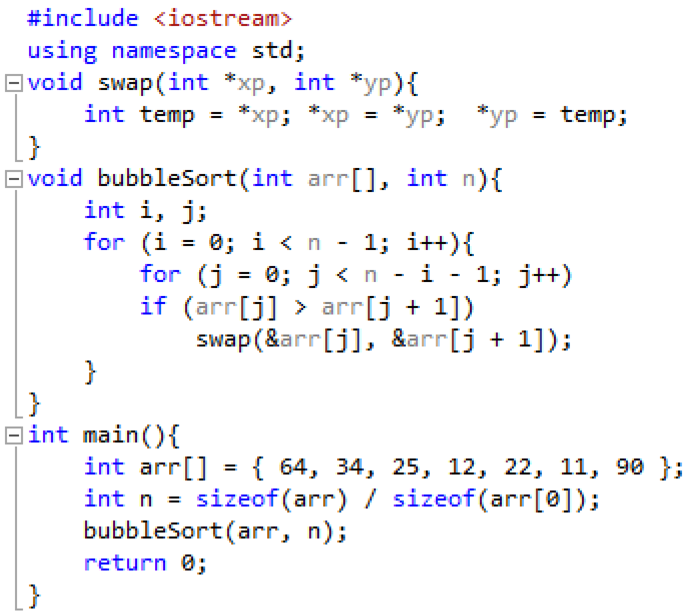
```cpp
#include <iostream>
using namespace std;
int search(int arr[], int n, int x){
    int i;
    for (i = 0; i < n; i++)
    if (arr[i] == x)
        return i;
    return -1;
}
int main(void){
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = search(arr, n, x);
    (result == -1) ? cout << "Element is not present in array"
        : cout << "Element is present at index " << result;
    return 0;
}
```

50) What is Bubble sort algorithm?

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

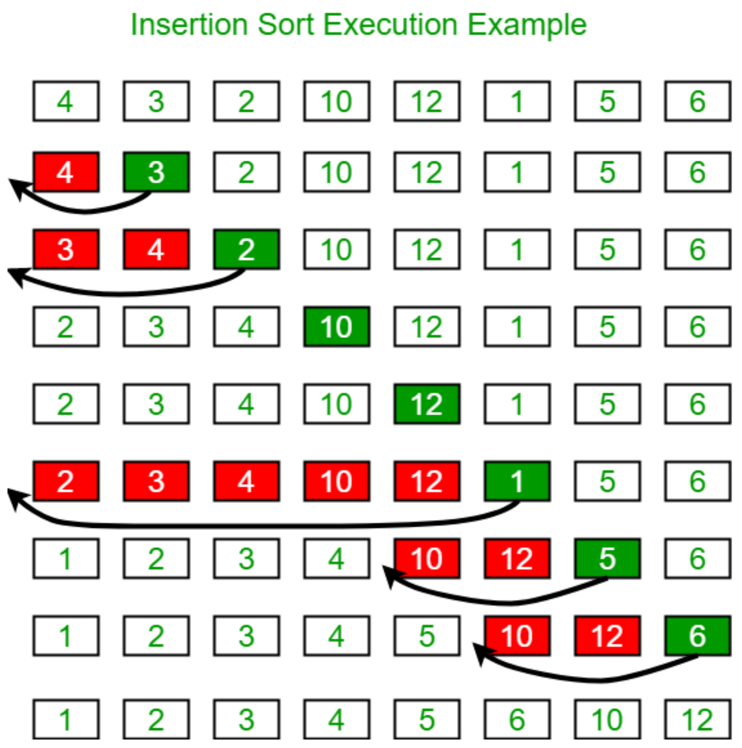


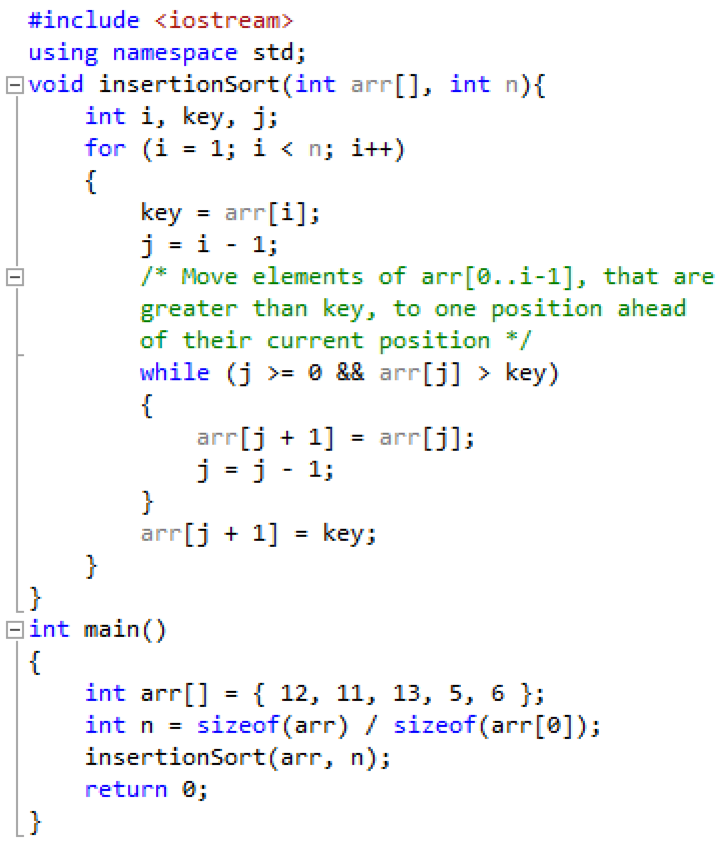
i = 0	j	0	1	2	3	4	5	6	7
	0	5	3	1	9	8	2	4	7
	1	3	5	1	9	8	2	4	7
	2	3	1	5	9	8	2	4	7
	3	3	1	5	9	8	2	4	7
	4	3	1	5	8	9	2	4	7
	5	3	1	5	8	2	9	4	7
	6	3	1	5	8	2	4	9	7
i = 1	0	3	1	5	8	2	4	7	9
	1	1	3	5	8	2	4	7	
	2	1	3	5	8	2	4	7	
	3	1	3	5	8	2	4	7	
	4	1	3	5	2	8	4	7	
	5	1	3	5	2	4	8	7	
i = 2	0	1	3	5	2	4	7	8	
	1	1	3	5	2	4	7		
	2	1	3	5	2	4	7		
	3	1	3	2	5	4	7		
	4	1	3	2	4	5	7		
i = 3	0	1	3	2	4	5	7		
	1	1	3	2	4	5			
	2	1	2	3	4	5			
	3	1	2	3	4	5			
i = 4	0	1	2	3	4	5			
	1	1	2	3	4				
	2	1	2	3	4				
i = 5	0	1	2	3	4				
	1	1	2	3					
i = 6	0	1	2	3					
		1	2						



```cpp
#include <iostream>
using namespace std;
void swap(int *xp, int *yp){
    int temp = *xp; *xp = *yp;  *yp = temp;
}
void bubbleSort(int arr[], int n){
    int i, j;
    for (i = 0; i < n - 1; i++){
        for (j = 0; j < n - i - 1; j++)
        if (arr[j] > arr[j + 1])
            swap(&arr[j], &arr[j + 1]);
    }
}
int main(){
    int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    return 0;
}
```

50) What is Insertion Sort algorithm?

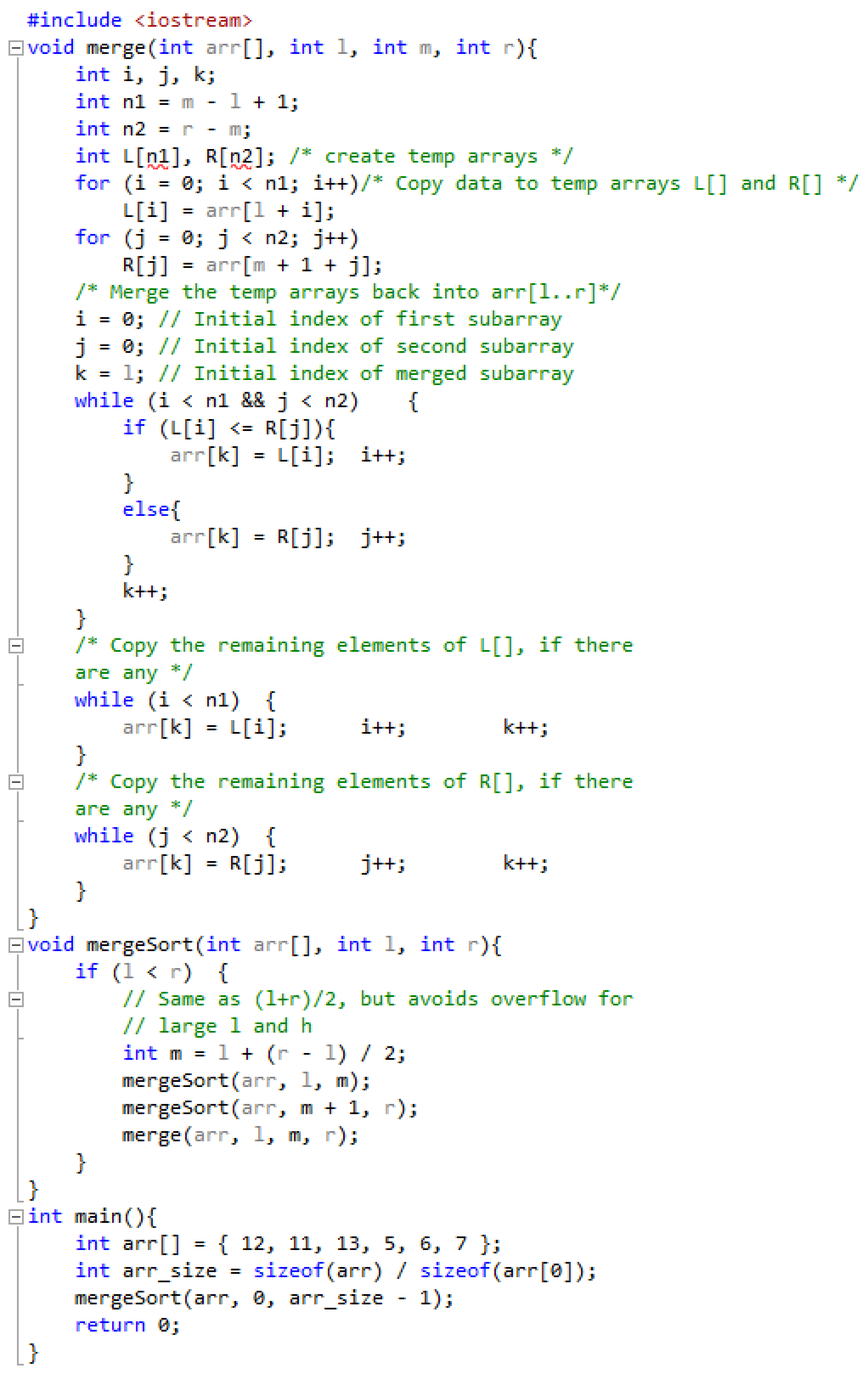




```
#include <iostream>
using namespace std;
void insertionSort(int arr[], int n){
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i - 1;
        /* Move elements of arr[0..i-1], that are
        greater than key, to one position ahead
        of their current position */
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main()
{
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    return 0;
}
```

50) What is Merge sort algorithm?

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(n log n), it is one of the most respected algorithms. Merge sort first divides the array into equal halves and then combines them in a sorted manner.



```
#include <iostream>
void merge(int arr[], int l, int m, int r){
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2]; /* create temp arrays */
    for (i = 0; i < n1; i++)/* Copy data to temp arrays L[] and R[] */
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)    {
        if (L[i] <= R[j]){
            arr[k] = L[i];  i++;
        }
        else{
            arr[k] = R[j];  j++;
        }
        k++;
    }
    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1)  {
        arr[k] = L[i];      i++;        k++;
    }
    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2)  {
        arr[k] = R[j];      j++;        k++;
    }
}
void mergeSort(int arr[], int l, int r){
    if (l < r)  {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
int main(){
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, arr_size - 1);
    return 0;
}
```

50) What is Quick sort algorithm?

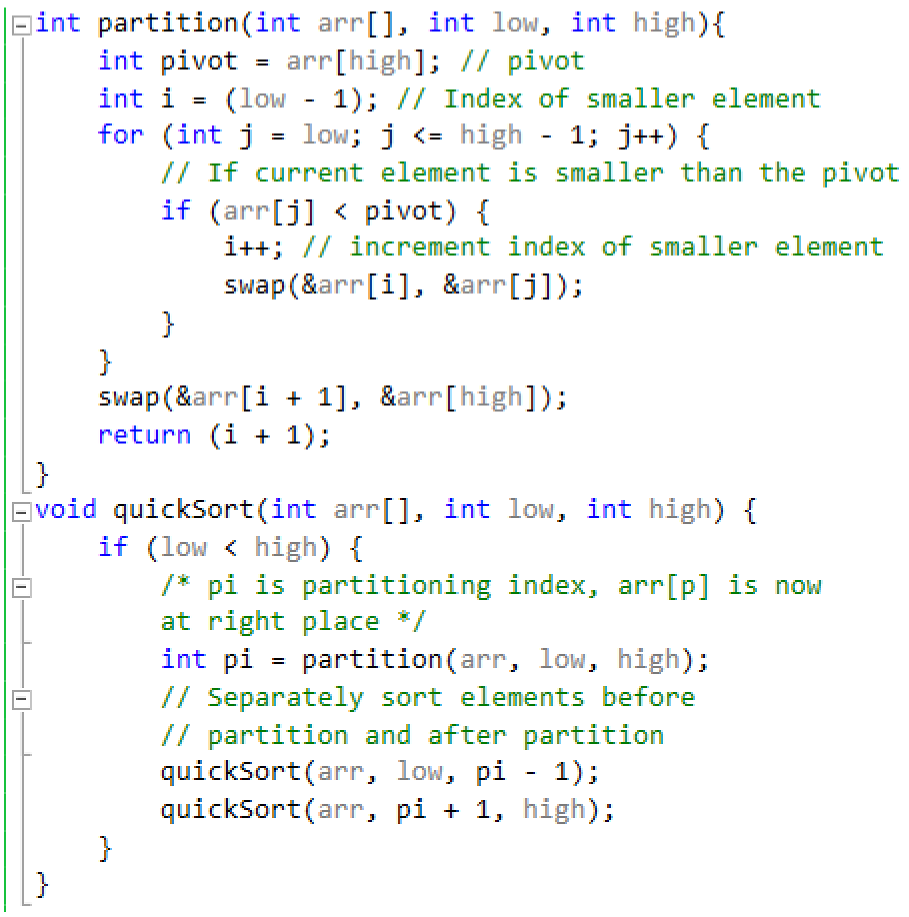
Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of QuickSort that pick pivot in different ways.

1). Always pick first element as pivot.

2). Always pick last element as pivot (implemented below)

3). Pick a random element as pivot.

4). Pick median as pivot.



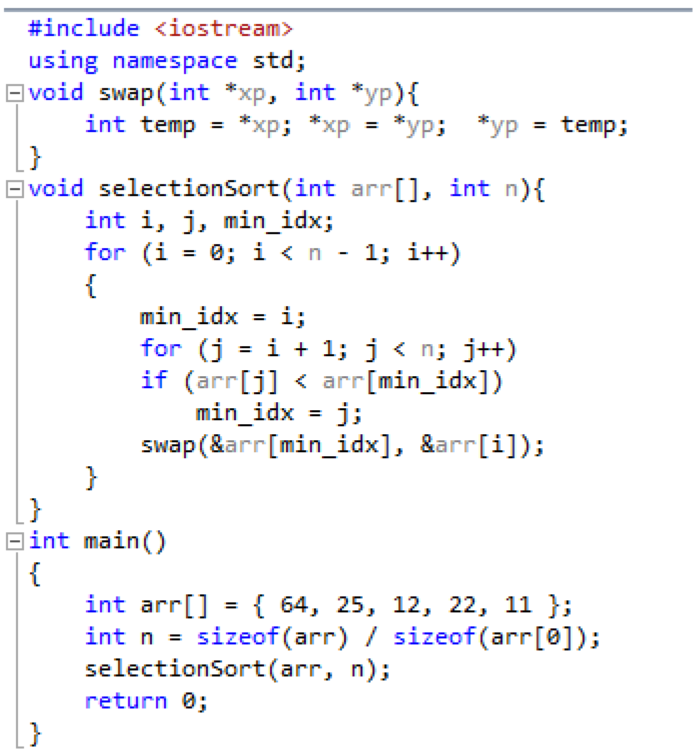
```
int partition(int arr[], int low, int high){
    int pivot = arr[high]; // pivot
    int i = (low - 1); // Index of smaller element
    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than the pivot
        if (arr[j] < pivot) {
            i++; // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        /* pi is partitioning index, arr[p] is now
        at right place */
        int pi = partition(arr, low, high);
        // Separately sort elements before
        // partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

50) What is Selection sort algorithm?

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.



```
#include <iostream>
using namespace std;
void swap(int *xp, int *yp){
    int temp = *xp; *xp = *yp;  *yp = temp;
}
void selectionSort(int arr[], int n){
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++)
    {
        min_idx = i;
        for (j = i + 1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;
        swap(&arr[min_idx], &arr[i]);
    }
}
int main()
{
    int arr[] = { 64, 25, 12, 22, 11 };
    int n = sizeof(arr) / sizeof(arr[0]);
    selectionSort(arr, n);
    return 0;
}
```

1. What is data structure?
Ans: The logical and mathematical model of a particular organization of data is called data structure. There are two types of data structure
i) Linear ii) Nonlinear

2. What are the goals of Data Structure?
Ans: It must rich enough in structure to reflect the actual relationship of data in real world. The structure should be simple enough for efficient processing of data. 1) To design an algorithm that is easy to understand, code and debugging. 2) To design an algorithm that Make efficient use of commuter recourses.

3. What does abstract Data Type Mean?
Ans: Data type is a collection of values and a set of operations on these values. Abstract data type refer to the mathematical concept that define the data type.
It is a useful tool for specifying the logical properties of a data type.
ADT consists of two parts
1) Values definition 2) Operation definition
Example:-The value definition for the ADT RATIONAL states that RATIONAL value consists of two integers, second doesn't equal to zero.
The operator definition for ADT RATIONAL includes the operation of creation (make rational) addition, multiplication and test for equality.

4. What is the difference between a Stack and an Array?
Ans:
i) Stack is a ordered collection of items
ii) Stack is a dynamic object whose size is constantly changing as items are pushed and popped .
iii) Stack may contain different data types
iv) Stack is declared as a structure containing an array to hold the element of the stack, and an integer to indicate the current stack top within the array.
ARRAY

i) Array is an ordered collection of items
ii) Array is a static object i.e. no of item is fixed and is assigned by the declaration of the array
iii) It contains same data types.
iv) Array can be home of a stack i.e. array can be declared large enough for maximum size of the stack.

5. What do you mean by recursive definition?
Ans: The definition which defines an object in terms of simpler cases of itself is called recursive definition.

6. What is sequential search?
Ans: In sequential search each item in the array is compared with the item being searched until a match occurs. It is applicable to a table organized either as an array or as a linked list.

7. What actions are performed when a function is called?
Ans: When a function is called
i) arguments are passed
ii) local variables are allocated and initialized
ii) transferring control to the function

8. What actions are performed when a function returns?
Ans:
i) Return address is retrieved
ii) Function's data area is freed
iii) Branch is taken to the return address

9. What is a linked list?
Ans: A linked list is a linear collection of data elements, called nodes, where the linear order is given by pointers. Each node has two parts first part contain the information of the element second part contains the address of the next node in the list.

10. What are the advantages of linked list over array (static data structure)?
Ans:
The disadvantages of array are
i) unlike linked list it is expensive to insert and delete elements in the array
ii) One can't double or triple the size of array as it occupies block of memory space.

In linked list
i) each element in list contains a field, called a link or pointer which contains the address of the next element
ii) Succes

11. Can we apply binary search algorithm to a sorted linked list, why?
Ans: No.
Reason: Binary search algorithm is based on the logic of reducing your input size by half in every step until your search succeeds or input gets exhausted. Important point here is **"the step to reduce input size should take constant time".** In case of an array, it's always a simple comparison based on array indexes that takes O(1) time.
But in case of Linked list you don't have indexes to access items. To perform any operation on a list item, you first have to reach it by traversing all items before it. So to divide list by half you first have to reach middle of the list then perform a comparison. Getting to middle of list takes O(n/2)[you have to

traverse half of the list items] and comparison takes O(1).
Total = O(n/2) + O(1) = O(n/2)

So the input reduction step does not take constant time. It depends on list size. hence violates the essential requirement of Binary search.

---

12. What do you mean by free pool?
Ans: Pool is a list consisting of unused memory cells which has its own pointer.

13. What do you mean by garbage collection?
Ans: It is a technique in which the operating system periodically collects all the deleted space onto the free storage list.
It takes place when there is minimum amount of space left in storage list or when CPU is ideal.
The alternate method to this is to immediately reinsert the space into free storage list which is time consuming.

14. What do you mean by overflow and underflow?
Ans: When new data is to be inserted into the data structure but there is no available space i.e. free storage list is empty this situation is called overflow.
When we want to delete data from a data structure that is empty this situation is called underflow.

15. What are the disadvantages array implementations of linked list?
Ans:
i) The no of nodes needed can't be predicted when the program is written.
ii) The no of nodes declared must remain allocated throughout its execution

16. What is a queue?
Ans: A queue is an ordered collection of items from which items may be deleted at one end (front end) and items inserted at the other end (rear end).
It obeys FIFO rule there is no limit to the number of elements a queue contains.

17. What is a priority queue?
Ans: The priority queue is a data structure in which the intrinsic ordering of the elements (numeric or alphabetic)
Determines the result of its basic operation. It is of two types
i) Ascending priority queue- Here smallest item can be removed (insertion is arbitrary)
ii) Descending priority queue- Here largest item can be removed (insertion is arbitrary)

18. What are the disadvantages of sequential storage?
Ans:
i) Fixed amount of storage remains allocated to the data structure even if it contains less element.
ii) No more than fixed amount of storage is allocated causing overflow

19. What are the disadvantages of representing a stack or queue by a linked list?
Ans:
i) A node in a linked list (info and next field) occupies more storage than a corresponding element in an array.
ii) Additional time spent in managing the available list.

20. What is dangling pointer and how to avoid it?
Ans: After a call to free(p) makes a subsequent reference to *p illegal, i.e. though the storage to p is

freed but the value of p(address) remain unchanged .so the object at that address may be used as the value of *p (i.e. there is no way to detect the illegality).Here p is called dangling pointer.
To avoid this it is better to set p to NULL after executing free(p).The null pointer value doesn’t reference a storage location it is a pointer that doesn’t point to anything.

21. What are the disadvantages of linear list?
Ans:
i) We cannot reach any of the nodes that precede node (p)
ii) If a list is traversed, the external pointer to the list must be persevered in order to reference the list again

22. Define circular list?
Ans: In linear list the next field of the last node contain a null pointer, when a next field in the last node contain a pointer back to the first node it is called circular list.
Advantages – From any point in the list it is possible to reach at any other point

23. What are the disadvantages of circular list?
Ans:
i) We can’t traverse the list backward
ii) If a pointer to a node is given we cannot delete the node

24. Define double linked list?
Ans: It is a collection of data elements called nodes, where each node is divided into three parts
i) An info field that contains the information stored in the node
ii) Left field that contain pointer to node on left side
iii) Right field that contain pointer to node on right side

25. Is it necessary to sort a file before searching a particular item ?
Ans:
If less work is involved in searching a element than to sort and then extract, then we don’t go for sort
If frequent use of the file is required for the purpose of retrieving specific element, it is more efficient to sort the file.
Thus it depends on situation.

26. What are the issues that hamper the efficiency in sorting a file?
Ans: The issues are
i) Length of time required by the programmer in coding a particular sorting program
ii) Amount of machine time necessary for running the particular program
iii)The amount of space necessary for the particular program .

27. Calculate the efficiency of sequential search?
Ans: The number of comparisons depends on where the record with the argument key appears in the table
If it appears at first position then one comparison
If it appears at last position then n comparisons
Average=(n+1)/2 comparisons
Unsuccessful search n comparisons
Number of comparisons in any case is O (n).

28. Is any implicit arguments are passed to a function when it is called?
Ans: Yes there is a set of implicit arguments that contain information necessary for the function to execute and return correctly. One of them is return address which is stored within the function’s data

area, at the time of returning to calling program the address is retrieved and the function branches to that location.

29. Parenthesis is never required in Postfix or Prefix expressions, why?
Ans: Parenthesis is not required because the order of the operators in the postfix /prefix expressions determines the actual order of operations in evaluating the expression

30. List out the areas in which data structures are applied extensively?
Ans:
Compiler Design,
Operating System,
Database Management System,
Statistical analysis package,
Numerical Analysis,
Graphics,
Artificial Intelligence,
Simulation

31. What are the major data structures used in the following areas: network data model & Hierarchical data model.
Ans:
RDBMS – Array (i.e. Array of structures)
Network data model – Graph
Hierarchical data model – Trees

32. If you are using C language to implement the heterogeneous linked list, what pointer type will you use?
Ans: The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

33. Minimum number of queues needed to implement the priority queue?
Ans: Two. One queue is used for actual storing of data and another for storing priorities.

34. What is the data structures used to perform recursion?
Ans: Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.
Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

35. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?
Ans: Polish and Reverse Polish notations.

36. Convert the expression ((A + B) * C – (D – E) ^ (F + G)) to equivalent Prefix and Postfix notations.
Ans: Prefix Notation:
^ – * +ABC – DE + FG
Postfix Notation:
AB + C * DE – – FG + ^

37. Sorting is not possible by using which of the following methods?
(a) Insertion
(b) Selection
(c) Exchange
(d) Deletion

Ans: (d) Deletion.
Using insertion we can perform insertion sort, using selection we can perform selection sort, using exchange we can perform the bubble sort (and other similar sorting methods). But no sorting method can be done just using deletion.

38. List out few of the Application of tree data-structure?
Ans:
The manipulation of Arithmetic expression,
Symbol Table construction,
Syntax analysis.

39. List out few of the applications that make use of Multilinked Structures?
Ans: Sparse matrix, Index generation.

40. in tree construction which is the suitable efficient data structure?
(A) Array (b) Linked list (c) Stack (d) Queue (e) none
Ans: (b) Linked list

41. What is the type of the algorithm used in solving the 8 Queens problem?
Ans: Backtracking

42. In an AVL tree, at what condition the balancing is to be done?
Ans: If the ‘pivotal value’ (or the ‘Height factor’) is greater than 1 or less than –1.

43. There are 8, 15, 13, 14 nodes were there in 4 different trees. Which of them could have formed a full binary tree?
Ans: 15
In general:
There are 2n-1 nodes in a full binary tree.

By the method of elimination:
Full binary trees contain odd number of nodes. So there cannot be full binary trees with 8 or 14 nodes, so rejected. With 13 nodes you can form a complete binary tree but not a full binary tree. So the correct answer is 15.
Note: Full and Complete binary trees are different. All full binary trees are complete binary trees but not vice versa.

44. In RDBMS, what is the efficient data structure used in the internal storage representation?
Ans: B+ tree. Because in B+ tree, all the data is stored only in leaf nodes, that makes searching easier. This corresponds to the records that shall
be stored in leaf nodes.

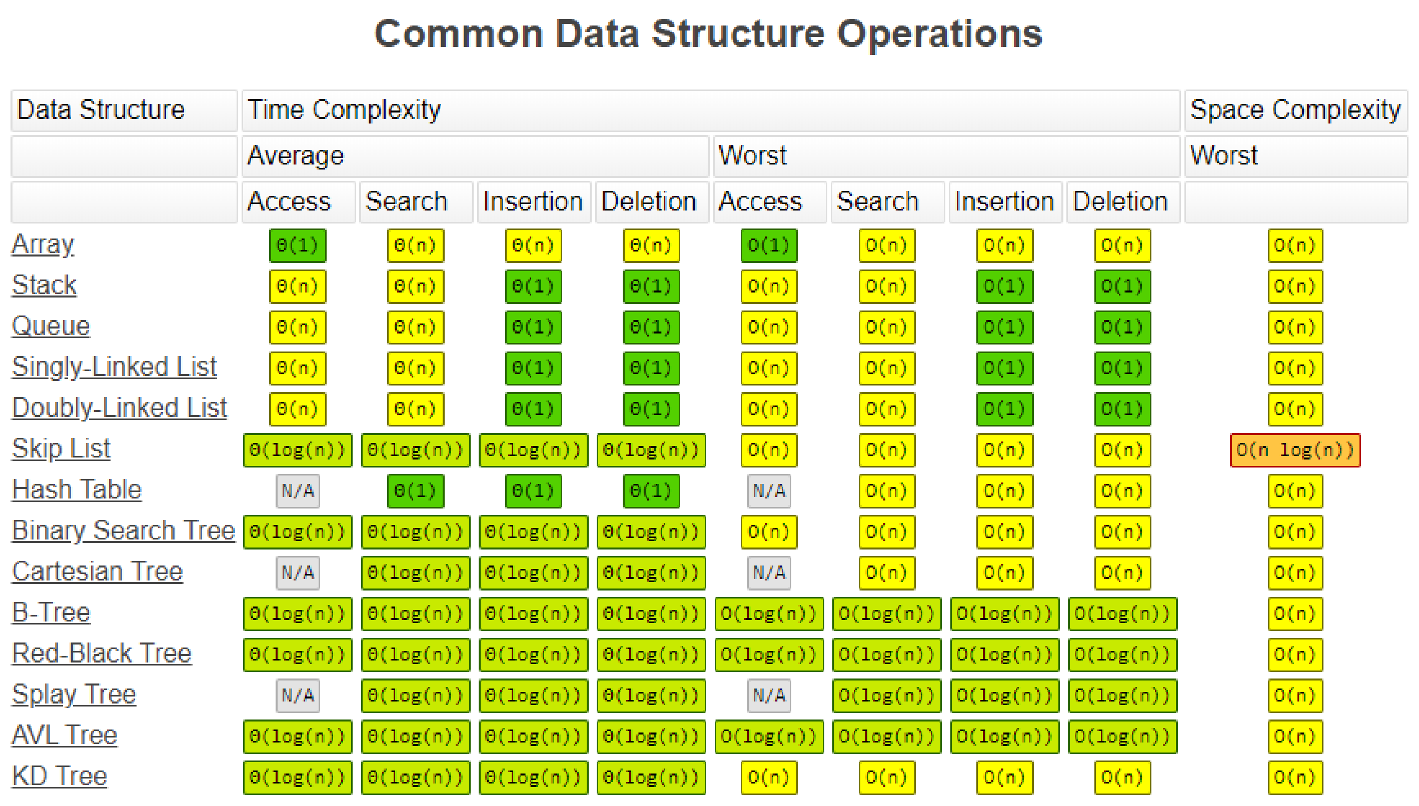
45. One of the following tree structures, which is, efficient considering space and time complexities?
a) Incomplete Binary Tree. b) Complete Binary Tree. c) Full Binary Tree.

Ans:
b) Complete Binary Tree.
By the method of elimination:
Full binary tree loses its nature when operations of insertions and deletions are done. For incomplete binary trees,
extra property of complete binary tree is maintained even after operations like additions and deletions are done on it.

46. What is a spanning Tree?
Ans: A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

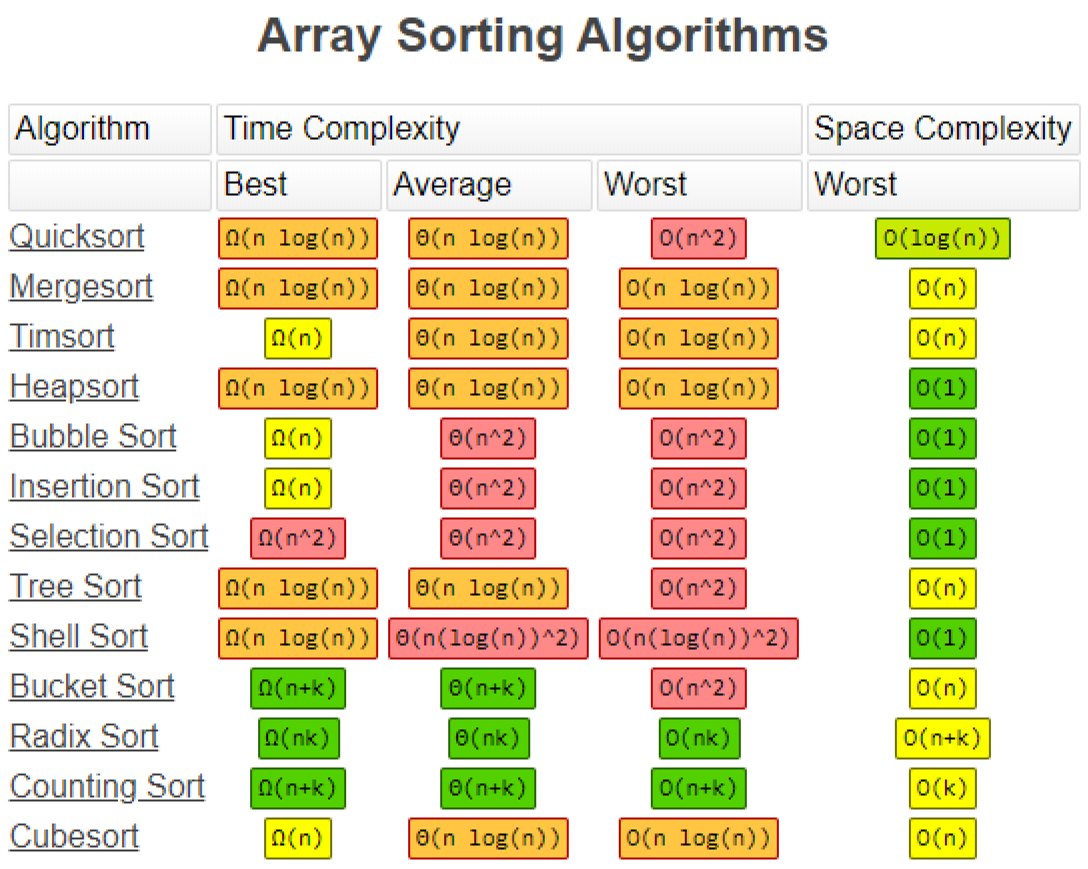
47. Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?
Ans: No.
Minimal spanning tree assures that the total weight of the tree is kept at its minimum. But it doesn't mean that the distance between any two nodes involved in the minimum-spanning tree is minimum.

48. Whether Linked List is linear or Non-linear data structure?
Ans: According to Storage Linked List is a Non-linear one.



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)



Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)