Example codes

// ID : 18CSE033

20

int out[n], count[10];

for(int i = 0; i < 10; i++) count[i] = 0;

ara[k++] = arr[j++];

int mi = i;

}

 $for(int j = i+1; j < n; j++){}$

1f(arr[m1] > arr[j])

mi = j;

arr[j+1] = temp;

void printArr(int arr[], int n)

12 13

14 15

16

17

19 20

21 22 }

23

25

6

8 9

10 11

12

13

14

15

16

17 18

23 24 #include <iostream> using namespace std;

void bubbleSort(int arr[], int n)

void printArr(int arr[], int n)

65_Deletion_from_a_Linked_List_at_any_position.cpp *

if(index < 0)(

bool insertHode(int data, int index)

Node* t - new Node(data);

Node* prev = first; for(int i = 1; i < index && prev; i++) prev = prev->next;

t->next = prev->next;

if(index -- 0) {
 t->next - first;
 first - t;
 return true;

if(prev){

int deleteNode(int index) if (index < 0) {
 cout << "inde

05 Deletion from a Linked List at any position.cpp ×

cout << "index must be grater than or equal to 0." << endl;

cout << "Linked List is not long enough to insert node at index: " << index << endl;

62 63

100 101

109 110 111

114

117 118

120

15

16

17 18

19

20 21 22

23

24 25

26

27

{

if(item < p->data)
 p->l = deleteNode(p->l, item); p if(item > p->datm)
p->r = deleteNode(p->r, item);

struct Node *q;
if(height(p->1) > height(p->r))
{

p->data = q->data; p->l = deleteMode(p->l, q->data);

q = insuc(p->r); p->data = q->data; p->r = deleteNode(p->r, q->data);

q = inpre(p->1);

int dist[V]; // Output/retuslt array
void Dijkstro(int graph[V][V], int start)

}while(i < j);</pre>

return j;

if(1 < h) {

swap(arr[1], arr[j]);

void quickSort(int arr[], int 1, int h)

quickSort(arr, 1, p);

quickSort(arr, p+1, h);

int p = partition(arr, 1, h);

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

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

swap(arr[j+1], arr[j]);

if(arr[j+1] < arr[j]){

swpped = true;

if(swpped == false) break;

bool swpped = false;

// C++ program for implementation of Depth-first search

Sunday, November 21, 2021 12:48 AM

```
× 17_057-qpp
using namespace std;
void BFS(int arr[][7], int node, int n)
       int visited[n] = {0};
      queue<int> q;
       int v = node;
      printf( %d , v);
visited[v] = 1;
      q.push(v);
      while([q.empty())
             v = q.front();
           v = q.fronc();
q.pop();
for(int i = 1; i < n; i++){
    if(arr[v][i] == 1 && visited[i] == 0){
        printf("%d ", i);
        visited[i] = 1;
        c.push(i);</pre>
       printf("\n");
int main()
H.USωp x V 17,051.αpp
// Name : Md Noor E Musa
```

```
#include <iostream>
using namespace std;
void DFS(int arr[][7], int node, int n)
     static int visited[7] = {0};
if(visited[node] == 0){
   printf("%d ", node);
   visited[node] = 1;
           for(int i = 1; i < n; i++)
                 if(arr[node][i] && visited[i] == 0)
                      DFS(arr, i, n);
     }
int main()
      int g[7][7] - {
           {0, 0, 0, 0, 0, 0, 0},
           {0, 0, 1, 1, 0, 0, 0},
{0, 1, 0, 0, 1, 0, 0},
{0, 1, 0, 0, 1, 0, 0},
        01_Bussic_Sortupp × ▼ 05_Quick_Sortupp × ▼ 13_Heap_Sortupp × ▼ 16_Insertion_Sortupp × ▼ 19_Socialin_Sortupp × ▼ 20_Morge_Sortupp × ▼ 21_Mortus_Sortupp ×
       )
 18
       void countingSort(int arr[], int n, int p) {
```

```
23
24
25
          for(int i = 0; i < n; i++) count[(arr[i]/p)%10]++;
         for(int i = 1; i < 10; i++) count[i] + count[i-1];
27
28
          for(int i = n-1; i >= 0; i--){}
29
30
             out[count[(arr[i]/p)%10]-1] = arr[i];
             count[(arr[i]/p)%10]--;
          for(int i = 0; i < n; i++) arr[i] = out[i];
34
     1
     void radixsort(int arr[], int n) {
         int mx - getMax(arr, n);
          for(int p = 1; mx/p>0; p*-10){
39
             countingSort(arr, n, p);
40
    )
    void printArr(int arr[], int n)
         roie,Sartapp x 🔻 de Quick,Sartapp x 🔻 13. Heap,Sartapp x 🔻 18. Invention,Sartapp x 🔻 19. Selection,Sartapp x 🔻 20. Marge,Sartapp x
     void merge(int arr[], int 1, int mid, int h)
10
    1
         int i = 1, j = mid+1, k = 1, ara[h+1];
          while(i <= mid && j <= h) {
              if(arr[i] < arr[j])</pre>
                 ara[k++] = arr[i++];
```

```
while(i <= mid) ara[k++] = arr[i++];
while(j <= h) ara[k++] = arr[j++];
20
23
24
          while(1 <= h){
              arr[1] = ara[1];
              1++;
     1
     void mergeSort(int arr[], int 1, int h)
30
         if(1 < h) {
             int m = 1 + (h-1)/2;
             mergeSort(arr, 1, m);
34
              mergeSort(arr, m+1, h);
             merge(arr, 1, m, h);
36
    81 Buttle, Sertupp x V 81, Quilde, Sertupp x V 13, Heap, Sortupp x V 18, Insertion, Sertupp x V 19, Selection Sertupp x V 20, Merge, Sertupp x V 21, Radio Sertupp
     // C++ program for implementation of Selection Sort
 4
     #include <iostream>
     using namespace std;
 8
     void selectionSort(int arr[], int n)
 9
10
     {
11
           for(int i = 0; i < n-1; i++){
```

```
18
              if(mi != i)
19
20
                   swap(arr[mi], arr[i]);
21
22
23
    void printArr(int arr[], int n)
24
25
    OI, Bubble, Sortupp x V 09, Quick, Sortupp x V 13, Heap, Sortupp x V 18, Heap for Lopp x V 18, Selection, Sortupp x V 20, Merge, Sortupp x V 21, Radia, Sortupp x
    // C++ program for implementation of Insertion Sort
 5
 6
    #include <iostream>
    using namespace std;
 8
    void insertionSort(int arr[], int n)
 9
10
11
         for(int i = 1; i < n; i++){
12
              int j = i-1;
              int temp = arr[i];
13
14
              while(j > -1 && temp < arr[j]){
15
                   arr[j+1] = arr[j];
16
17
                   j--;
18
```

```
Olfsiddic Sortopo X V Olfsidd, Sortopo X V 13 Mary Sortopo X V 18 Javater, Sortopo X V 19 Selector, Sortopo X V 25 Maryo, Sortopo X V 27 Roda, Sortopo X
    void heapify(int arr[], int n, int i)
9
10
    1
         int b = i, l = i*2+1, r = i*2+2;
11
13
         if(1 < n && arr[1] > arr[b])
14
             b = 1;
16
         if(r < n && arr[r] > arr[b])
             b = r;
18
         if(b != i){
19
             swap(arr[b], arr[i]);
20
             heapify(arr, n, b);
22
24
    void heapSort(int arr[], int n)
26
         for(int i = n/2-1; i >= 0; i--)
28
             heapify(arr, n, i);
29
         for(int i = n-1; i > 0; i--){
30
31
             swap(arr[0], arr[i]);
32
             heapify(arr, i, θ);
34
     // C++ program for implementation of Bubble Sort
```

```
05 Detetion from a United List at any positionapp H
           (nt deleteNode(int index)
63
64
65
                if(index < 8)(
cout << "index must be grater than or equal to 0." << endl;
                     exit(1);
                int ele = -1;
                Node *p - first, *q - nullptr;
                 if(index -= 0){
   first = first >next;
   ele = p >data;
70
71
72
73
74
75
76
77
                     p = nullptr;
78
79
                     for(tet 1 - 0; 1 < index 85 p; 1++){
                          q-p;
p-p->mext;
81
82
                     if(p)[
                          q->next = p->next;
                          ele = p->data;
                          p = nullptr;
88
89
98
91
92
                          cout << "Linked List is not long enough to delete node at index: " << index << endl;
                          exit(1);
                 return ele;
```

```
bool insertHode(int data, int index)
                 if (index < 0) (
cout << "index must be grater than or equal to 0." << endl;
35
38
39
                  Node *temp - new Hode(data);
41
42
                  if(index - 0)
44
45
                       temp->next = first;
                       first - temp;
return true;
48
49
58
                 Mode *prev - first;
for (int i - 1; i < index &A prev; i++)
                  if(prev)
                      temp->next = prev->next;
prev->next = temp;
57
58
59
                  cout << "Linked List is not long enough to insert node at index: " << index << endl;
62
            int deleteNode(int index)
       14. Huffman, Agorithm.cpc × V 11. Insversing Binary Tree.cpc × V 12. Strary Search, Tree.cpc × V 05. Deletion, from a Linked List, any position.cpc × struct Node* deleteNode(struct Node*p, int item)
90
            if(p -- MULL)
return MULL;
             #((p->1 && (p->r)
96
97
                 if(p == root)
                 root = MULL;
free(p);
return MULL;
```

```
14. Huffman, Algorithm.cop x V 11. Traversing, Binary, Tree.cop x V 12. Binary, Search, Tree.cop x V 05. Deterior, from a Linked List, at Juny, position.cop x
       struct Node* insertNode(struct Node *t, int data)
              if(t -- MILL)
18
                  t = (struct Node*)m
t->data = data;
t->l = t->r = NULL;
return t;
                                           e*)malloc(sizeof(struct Node));
              if(t->data < data)
             t->r = insertNode(t->r, data);
else if(t->data > data)
t->l = insertNode(t->l, data);
return t;
25
26
27
       void search(struct Node *t, int item)
32
33
34
             (*)
                  search(t->r, item);
else if(t->data > item);
search(t->l, item);
else
                    if(t->data ← item)
39
49
                         printf("Found %d\n", item);
              else printf("NOT Found %d\n", item);
45
```

14. Huffman, Agorithmusp x V 15. Shortest, Pathusp x V 11. Travesing, Binary, Treeusp x V 12. Binary, Search, Treeusp x V (5. Deletion, from a Linkest, List, any, position, up x

```
priority_queue(pi, vector(pi), greater(pi) > pq;
bool visited[V];
 20
 23
24
25
                                         for(int i = 0; i < V; i++){
    dist[i] = inf;
    visited[i] = false;</pre>
 26
27
28
29
30
31
                                        dist[start] = 0;
pq.push(make_pair(0, start));
                                            while(!pq.empty()) {
 32
33
34
                                                         pi node = pq.top();
pq.pop();
 35
36
                                                          visited[node.name] = true;
if(dist[node.name] < node.dist) continue;</pre>
                                                            for(int i = 0; i < V; i++) {
    if(graph[node.name][i] > 0 && visited[i] == false) {
        int new_dist = graph[node.name][i] + node.dist;
        if(new_dist < dist[i]) {</pre>
38
39
40
41
42
43
44
45
46
47
48
                                                                                                              dist[i] = new_dist;
                                                                                                               pq.push(make_pair(new_dist, i));
                                                                           1
                          Ot Babbic Sortago X V 09 Quick Sortago X V 18 Heap Sortago X V 18 
         9
                            int partition(int arr[], int 1, int h)
  10
                            {
                                                        int pivot = arr[1], i = 1, j = h;
  11
 12
  13
                                                                                   do { i++; } while(pivot >= arr[i]);
do { j--; } while(pivot < arr[j]);</pre>
  14
```