29/8/24

1.

```
#include <stdio.h>
int main() {
    int regNos[] = {101, 102, 103, 104, 105};
    int n = sizeof(regNos) / sizeof(regNos[0]);
    int search, found = 0;
    printf("Enter registration number to search: ");
    scanf("%d", &search);
    for (int i = 0; i < n; i++) {
        if (regNos[i] == search) {
            found = 1;
            break;
        }
    }
    if (found)
        printf("Registration number %d found.\n", search);
    else
        printf("Registration number %d not found.\n", search);

    return 0;
}
```

OUTPUT:

```
Enter registration number to search: 102
Registration number 102 found.


=== Code Execution Successful ===
```

2.

```
#include <stdio.h>
#include <string.h>
int checkNeedleInHaystack(const char *needle, const char *haystack) {
    for (int i = 0; needle[i]; i++) {
        if (!strchr(haystack, needle[i])) return 0;
    }
    return 1;
}
int main() {
    const char *needle = "abc";
    const char *haystack = "abcdef";
    printf("%d\n", checkNeedleInHaystack(needle, haystack));
    return 0;
}
```

OUTPUT:

```
1


=== Code Execution Successful ===
```

3.

```
#include <stdio.h>
int main() {
    int n, i, j, count;
    printf("Input the number of elements to be stored in the array: ");
    scanf("%d", &n);
    int arr[n], freq[n];
    printf("Input %d elements in the array:\n", n);
    for (i = 0; i < n; i++) {
```

```
        printf("element - %d : ", i);
        scanf("%d", &arr[i]);
        freq[i] = -1;
    }
    for (i = 0; i < n; i++) {
        count = 1;
        if (freq[i] != 0) {
            for (j = i + 1; j < n; j++) {
                if (arr[i] == arr[j]) {
                    count++;
                    freq[j] = 0;
                }
            }
            freq[i] = count;
        }
    }
    printf("The frequency of all elements of an array:\n");
    for (i = 0; i < n; i++) {
        if (freq[i] != 0) {
            printf("%d occurs %d times\n", arr[i], freq[i]);
        }
    }
    return 0;
}
```

OUTPUT:

```
Input the number of elements to be stored in the array: 3
Input 3 elements in the array:
element - 0 : 25
element - 1 : 12
element - 2 : 43
The frequency of all elements of an array:
25 occurs 1 times
12 occurs 1 times
43 occurs 1 times


=== Code Execution Successful ===
```

4.

```
#include <stdio.h>
#include <limits.h>
#define V 5
int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    return min_index;
}
void primMST(int graph[V][V]) {
    int parent[V], key[V];
    int mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
```

```
        mstSet[i] = 0;
    }
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
    }
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d - %d \t%d\n", parent[i], i, graph[parent[i]][i]);
}
int main() {
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };
    primMST(graph);
    return 0;
}
```

OUTPUT:

```
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5


=== Code Execution Successful ===
```

5.

```
#include <stdio.h>
int main() {
    int n, i, even[50], odd[50], e_count = 0, o_count = 0;
    printf("Input the number of elements to be stored in the array: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        int num;
        printf("element - %d : ", i);
        scanf("%d", &num);
        if (num % 2 == 0) even[e_count++] = num;
        else odd[o_count++] = num;
    }
    printf("The Even elements are:\n");
    for (i = 0; i < e_count; i++) printf("%d ", even[i]);
    printf("\nThe Odd elements are:\n");
    for (i = 0; i < o_count; i++) printf("%d ", odd[i]);
    return 0;
}
```

OUTPUT:

```
Input the number of elements to be stored in the array: 5
element - 0 : 25
element - 1 : 47
element - 2 : 42
element - 3 : 56
element - 4 : 32
The Even elements are:
42 56 32
The Odd elements are:
25 47

=== Code Execution Successful ===
```

6.

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

typedef struct Node {
    int vertex;
    struct Node* next;
} Node;

Node* adjList[MAX];
int visited[MAX];
int parent[MAX];

void addEdge(int src, int dest) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->vertex = dest;
    newNode->next = adjList[src];
    adjList[src] = newNode;
```

```
}

void dfs(int vertex, int target) {
    visited[vertex] = 1;
    if (vertex == target) {
        printf("Path: ");
        for (int v = target; v != -1; v = parent[v])
            printf("%d ", v);
        printf("\n");
        return;
    }
    for (Node* temp = adjList[vertex]; temp; temp = temp->next) {
        if (!visited[temp->vertex]) {
            parent[temp->vertex] = vertex;
            dfs(temp->vertex, target);
        }
    }
}

int main() {
    int edges, src, dest, start, target;
    printf("Enter number of edges: ");
    scanf("%d", &edges);
    for (int i = 0; i < edges; i++) {
        printf("Enter edge (src dest): ");
        scanf("%d %d", &src, &dest);
        addEdge(src, dest);
    }
    printf("Enter start and target vertices: ");
    scanf("%d %d", &start, &target);
```

```
    for (int i = 0; i < MAX; i++) {
        visited[i] = 0;
        parent[i] = -1;
    }
    dfs(start, target);
    return 0;
}
```

OUTPUT:

```
Enter number of edges: 3
Enter edge (src dest): 0 1
Enter edge (src dest): 2 3
Enter edge (src dest): 3 4
Enter start and target vertices: 0 4


=== Code Execution Successful ===
```

7.

```
#include <stdio.h>
int fibonacci(int n) {
    return (n <= 1) ? n : fibonacci(n - 1) + fibonacci(n - 2);
}
int sumFibonacci(int n) {
    return (n == 0) ? 0 : sumFibonacci(n - 1) + fibonacci(n);
}
int main() {
    int n = 1;
    printf("Sum of Fibonacci series up to %d: %d\n", n, sumFibonacci(n));
    return 0;
}
```

OUTPUT:

```
Sum of Fibonacci series up to 1: 1


=== Code Execution Successful ===
```

8.

```
#include <stdio.h>
void heapify(int arr[], int n, int i) {
    int largest = i, left = 2 * i + 1, right = 2 * i + 2;
    if (left < n && arr[left] > arr[largest]) largest = left;
    if (right < n && arr[right] > arr[largest]) largest = right;
    if (largest != i) {
        int temp = arr[i]; arr[i] = arr[largest]; arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);
    for (int i = n - 1; i > 0; i--) {
        int temp = arr[0]; arr[0] = arr[i]; arr[i] = temp;
        heapify(arr, i, 0);
    }
}
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
```

```
    int n = sizeof(arr) / sizeof(arr[0]);
    heapSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

OUTPUT:

```
5 6 7 11 12 13


=== Code Execution Successful ===
```

9.

```
#include <stdio.h>
int factorial(int n) {
    return (n == 0) ? 1 : n * factorial(n - 1);
}
int main() {
    int num;
    scanf("%d", &num);
    printf("Factorial of %d = %d\n", num, factorial(num));
    return 0;
}
```

OUTPUT:

```
6
Factorial of 6 = 720


=== Code Execution Successful ===
```

10.

```
#include <stdio.h>

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = arr[high], i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        quickSort(arr, low, i);
        quickSort(arr, i + 2, high);
    }
}
```

```
int main() {
    int n = 10, arr[10];
    printf("How many elements are you going to enter?: %d\n", n);
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) scanf("%d", &arr[i]);
    quickSort(arr, 0, n - 1);
    printf("Order of Sorted elements: ");
    for (int i = 0; i < n; i++) printf("%d ", arr[i]);
    return 0;
}
```

OUTPUT:

```
How many elements are you going to enter?: 10
Enter 10 elements: 2 3 5 7 1 9 3 8 0 4
Order of Sorted elements: 0 1 2 3 3 4 5 7 8 9

=== Code Execution Successful ===
```