Shubham Sharma
2021-22BCSE046

Week 2

Searching Algorithms
- Linear Search
- Binary Search

1. Given an unsorted array of elements and a key value, write a function to determine if the key exists in the array.

```
#include<iostream>
using namespace std;
int main(){
   cout<<"enter the size of array :"<<endl;
   int n;
   cin>>n;
   int arr[n];
   cout<<"enter the elements of array:"<<endl;
   for(int i =0; i < n ; i++){
      cin>>arr[i];
   }
   int k ;
   cout<<"enter the element to be found "<<endl;
   cin>>k;
   bool flag =0;
   for(int i =0; i< n ; i++){
      if(arr[i]==k){
         flag = 1;
         break;
      }
   }
   if(flag ==1){
      cout<<"Elements is present"<<endl;
   }else{
      cout<<"Element is not present"<<endl;
   }
}
```

output:

```
enter the size of array : 5
enter the elements of array: 1 2 3 4 5
enter the element to be found : 3
Elements is present
```

Shubham Sharma
2021-22BCSE046

2. Given a sorted array of elements and a key value, write a function to determine if the key exists in the array.

```
#include<iostream>
using namespace std;
int main(){
   cout<<"enter the size of array :"<<endl;
   int n;
   cin>>n;
   int arr[n];
   cout<<"enter the elements of array:"<<endl;
   for(int i =0; i < n ; i++){
      cin>>arr[i];
   }
   int k ;
   cout<<"enter the element to be found "<<endl;
   cin>>k;
   sort(arr, arr + n);
   bool flag =0;
   int s =0;
   int e = n-1;
   while(s<=e){
      int mid = s + (e-s)/2;
      if(arr[mid]==k){
         flag =1;
         break;
      }
      else if( arr[mid] > k){
         e= mid-1;
      }
      else{
         s = mid+1;
      }
   }
   if(flag ==1){
      cout<<"Elements is present"<<endl;
   }else{
      cout<<"Element is not present"<<endl;
   }
}
```

output:

```
enter the size of array : 5
enter the elements of array: 1 2 3 4 5
enter the element to be found: 2
Elements is present
```

Shubham Sharma
2021-22BCSE046

3. Design a program to manage employee records using structures in C++. The program should provide the following functionalities:
   - Add new employee records with the following details: Employee ID, Name, Department, Designation, DoB, DoJ, Salary
   - Display all employee records.
   - Display all the employees who have joined before/after a specific Date.
   - Search for an employee by their ID and display their details.
   - Update an employee's salary.
   - Delete an employee record by their ID.
   - Calculate and display the following:
   - Total number of employees
   - Average salary
   - Highest and lowest salaries

   Ensure that the program implements proper error handling and validation for user inputs. Use appropriate data structures and algorithms to manage and manipulate employee records efficiently.

```
#include<iostream>
using namespace std;

struct Employee {
   int employeeID;
   string name;
   string department;
   string designation;
   string dateOfBirth;
   string dateOfJoining;
   double salary;
};
vector<Employee> employees;

void addEmployee() {
   Employee emp;
   cout << "Enter Employee ID: ";
   cin >> emp.employeeID;
   cin.ignore();
   cout << "Enter Name: ";
   getline(cin, emp.name);
   cout << "Enter Department: ";
   getline(cin, emp.department);
   cout << "Enter Designation: ";
   getline(cin, emp.designation);
   cout << "Enter Date of Birth (YYYY-MM-DD): ";
   getline(cin, emp.dateOfBirth);
   cout << "Enter Date of Joining (YYYY-MM-DD): “;
   getline(cin, emp.dateOfJoining);
   cout << "Enter Salary: ";
   cin >> emp.salary;
```

Shubham Sharma
2021-22BCSE046

```
    employees.push_back(emp);
    cout << "Employee added successfully." << endl;
}
void displayAllEmployees() {
    cout << "Employee Records:" << endl;
    for (const auto& emp : employees) {
        cout << "Employee ID: " << emp.employeeID << endl;
        cout << "Name: " << emp.name << endl;
        cout << "Department: " << emp.department << endl;
        cout << "Designation: " << emp.designation << endl;
        cout << "Date of Birth: " << emp.dateOfBirth << endl;
        cout << "Date of Joining: " << emp.dateOfJoining << endl;
        cout << "Salary: " << emp.salary << endl << endl;
    }
}

void displayEmployeesByDate(bool before) {
    string date;
    cout << "Enter the specific date (YYYY-MM-DD): ";
    cin >> date;

    cout << "Employees joined ";
    if (before)
        cout << "before ";
    else
        cout << "after ";
    cout << date << ":" << endl;

    for (const auto& emp : employees) {
        if (before && emp.dateOfJoining < date)
            cout << emp.name << endl;
        else if (!before && emp.dateOfJoining > date)
            cout << emp.name << endl;
    }
}

void searchEmployeeByID() {
    int id;
    cout << "Enter Employee ID: ";
    cin >> id;

    auto it = find_if(employees.begin(), employees.end(), [id](const Employee& emp)
{
        return emp.employeeID == id;
    });

    if (it != employees.end()) {
        cout << "Employee found:" << endl;
```

output:

Shubham Sharma
2021-22BCSE046

```
        cout << "Name: " << it->name << endl;
        cout << "Department: " << it->department << endl;
        cout << "Designation: " << it->designation << endl;
        cout << "Date of Birth: " << it->dateOfBirth << endl;
        cout << "Date of Joining: " << it->dateOfJoining << endl;
        cout << "Salary: " << it->salary << endl;
    } else {
        cout << "Employee not found." << endl;
    }
}

void updateSalary() {
    int id;
    cout << "Enter Employee ID: ";
    cin >> id;

    auto it = find_if(employees.begin(), employees.end(), [id](const Employee&
emp) {
        return emp.employeeID == id;
    });

    if (it != employees.end()) {
        cout << "Enter new salary: ";
        cin >> it->salary;
        cout << "Salary updated successfully." << endl;
    } else {
        cout << "Employee not found." << endl;
    }
}

void deleteEmployee() {
    int id;
    cout << "Enter Employee ID: ";
    cin >> id;

    auto it = find_if(employees.begin(), employees.end(), [id](const Employee&
emp) {
        return emp.employeeID == id;
    });

    if (it != employees.end()) {
        employees.erase(it);
        cout << "Employee deleted successfully." << endl;
    } else {
        cout << "Employee not found." << endl;
    }
}

void calculateStatistics() {
    double totalSalary = 0.0;
    double highestSalary = -1.0;
```

```
    double lowestSalary = -1.0;
    for (const auto& emp : employees) {
        totalSalary += emp.salary;
        if (highestSalary == -1.0 || emp.salary > highestSalary)
            highestSalary = emp.salary;
        if (lowestSalary == -1.0 || emp.salary < lowestSalary)
            lowestSalary = emp.salary;
    }

    int totalEmployees = employees.size();
    double averageSalary = totalSalary / totalEmployees;

    cout << "Total number of employees: " << totalEmployees << endl;
    cout << "Average salary: " << averageSalary << endl;
    cout << "Highest salary: " << highestSalary << endl;
    cout << "Lowest salary: " << lowestSalary << endl;
}

int main() {
    int choice;
    do {
        cout << "Employee Management System" << endl;
        cout << "1. Add new employee" << endl;
        cout << "2. Display all employees" << endl;
        cout << "3. Display employees joined before a specific date" << endl;
        cout << "4. Display employees joined after a specific date" << endl;
        cout << "5. Search for an employee by ID" << endl;
        cout << "6. Update employee salary" << endl;
        cout << "7. Delete an employee" << endl;
        cout << "8. Calculate and display statistics" << endl;
        cout << "9. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                addEmployee();
                break;
            case 2:
                displayAllEmployees();
                break;
            case 3:
                displayEmployeesByDate(true);
                break;
            case 4:
                displayEmployeesByDate(false);
                break;
            case 5:
                searchEmployeeByID();
                break;
```

Shubham Sharma
2021-22BCSE046

```
            case 6:
                updateSalary();
                break;
            case 7:
                deleteEmployee();
                break;
            case 8:
                calculateStatistics();
                break;
            case 9:
                cout << "Exiting..." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 9);

    return 0;
}
```

Shubham Sharma
2021-22BCSE046

```
Employee Management System
1. Add new employee
2. Display all employees
3. Display employees joined before a specific date
4. Display employees joined after a specific date
5. Search for an employee by ID
6. Update employee salary
7. Delete an employee
8. Calculate and display statistics
9. Exit
Enter your choice: 1
Enter Employee ID: 22
Enter Name: john
Enter Department: cse
Enter Designation: student
Enter Date of Birth (YYYY-MM-DD): 2004-10-01
Enter Date of Joining (YYYY-MM-DD): 2022-07-01
Enter Salary: 0.0
Employee added successfully.
Employee Management System
1. Add new employee
2. Display all employees
3. Display employees joined before a specific date
4. Display employees joined after a specific date
5. Search for an employee by ID
6. Update employee salary
7. Delete an employee
8. Calculate and display statistics
9. Exit
Enter your choice: 2
Employee Records:
Employee ID: 22
Name: john
Department: cse
Designation: student
Date of Birth: 2004-10-01
Date of Joining: 2022-07-01
Salary: 0

Employee Management System
1. Add new employee
2. Display all employees
3. Display employees joined before a specific date
4. Display employees joined after a specific date
5. Search for an employee by ID
6. Update employee salary
7. Delete an employee
8. Calculate and display statistics
9. Exit
Enter your choice: 9
```

Shubham Sharma
2021-22BCSE046

4. Design a structure called Car with attributes such as car_ID, make, model, year, rental_price_per_day, from_date, and to_date. Write a program to allow users to input rental car data, store them in an array of structures, and then calculate and display the total rental cost for a particular car over a specified number of days.

```
#include<bits/stdc++.h>
using namespace std;
struct Car {
int car_ID;
string make;
string model;
int year;
double rental_price_per_day;
string from_date;
string to_date;
};
const int MAX_CARS = 100;
double calculateRentalCost(const Car& car, int numDays) {
return car.rental_price_per_day * numDays;
}
int main() {
Car cars[MAX_CARS];
int numCars;
cout << "Enter the number of rental cars: ";
cin >> numCars;
for (int i = 0; i < numCars; ++i) {
cout << "Enter Car ID: ";
cin >> cars[i].car_ID;
cout << "Enter Make: ";
cin >> cars[i].make;
cout << "Enter Model: ";
cin >> cars[i].model;
cout << "Enter Year: ";
cin >> cars[i].year;
cout << "Enter Rental Price per Day: ";
cin >> cars[i].rental_price_per_day;
cout << "Enter From Date (YYYY-MM-DD): ";
cin >> cars[i].from_date;
cout << "Enter To Date (YYYY-MM-DD): ";
cin >> cars[i].to_date;
}
int carID;
cout << "Enter Car ID to calculate rental cost: ";
cin >> carID;
Car selectedCar;
bool found = false;
for (int i = 0; i < numCars; ++i) {
        if (cars[i].car_ID == carID) {
                selectedCar = cars[i];
                found = true;
                break;
        }
```

Shubham Sharma
2021-22BCSE046

```
}
if (found) {
        int numDays;
        cout << "Enter number of rental days: ";
        cin >> numDays;
        double totalCost = calculateRentalCost(selectedCar, numDays);
        cout << "Total rental cost for Car ID " << selectedCar.car_ID << ": $" <<
        totalCost << endl;
} else {
        cout << "Car with ID " << carID << " not found." << endl;
}
return 0;
}
```

output:

```
Enter the number of rental cars: 1
Enter Car ID: 12
Enter Make: Tata
Enter Model: Nano
Enter Year: 2024
Enter Rental Price per Day: 100
Enter From Date (YYYY-MM-DD): 2024-02-05
Enter To Date (YYYY-MM-DD): 2024-02-10
Enter Car ID to calculate rental cost: 12
Enter number of rental days: 5
Total rental cost for Car ID 12: $500
```

5. Given a 2D array of integers where each row and column are sorted in ascending order, write a function to efficiently search for a target value in the array. Print the # of comparisons required before finding a target value through row-based and column-based searches.

```
#include <bits/stdc++.h>
using namespace std;
bool searchMatrix(vector<vector<int>>& matrix, int target){
int row = matrix.size();
int col = matrix[0].size();
int s = 0, e = row * col - 1;
while (s <= e){
        int mid = s + (e - s) / 2;
        if (matrix[mid / col][mid % col] == target){
        return true;
}
else if (matrix[mid / col][mid % col] > target){
        e = mid - 1;
      }
}
else{
      s = mid + 1;
      }
}
return false;
}
```

Shubham Sharma
2021-22BCSE046

```
int main(){
int row , col;
cout <<"Enter the number of rows and columns: ";
cin>> row>>col;
vector<vector<int>> matrix;
for (int i = 0; i < row; i++) {
        vector<int> temp;
        for (int j = 0; j < col; j++) {
        temp.push_back(i * 4 + j);
        }
        matrix.push_back(temp);
}

int k ;
cout<<"enter the element to be found: ";
cin>>k;
cout <<"key value present or not: "<<searchMatrix(matrix, k)<<endl;
}
```

output:

```
Enter the number of rows and columns: 3 3
enter the element to be found: 2
key value present or not: 1
```

6. Given a sorted array of integers and a target value, write efficient functions to find the first and last occurrences of the key value in the array.

Sample Array:

3	4	4	4	5	5	8	9
0	1	2	3	4	5	6	7

Key value: 4
Index of the first occurrence: 1
Index of the last occurrence: 3

```
#include <bits/stdc++.h>
using namespace std;
int main(){
int k;
cout << "Enter the number of elements: ";
cin >> k;
vector<int> nums(k);
cout << "Enter elements: ";
for (int i = 0; i < k; ++i){
cin >> nums[i];
}
vector<int> ans;
```

Shubham Sharma
2021-22BCSE046

```
int n = nums.size();
int first = -1;
int last = -1;
int s = 0;
int e = n - 1;
int target;
cout << "enter the target element: ";
cin >> target;
while (s <= e){
int mid = s + (e - s) / 2;
if (nums[mid] > target){
e = mid - 1;
}
else if (nums[mid] < target){
s = mid + 1;
}
else{
last = mid;
s = mid + 1;
}
}
s = 0;
e = n - 1;
while (s <= e){
int mid = s + (e - s) / 2;
if (nums[mid] > target){
e = mid - 1;
}
else if (nums[mid] < target){
s = mid + 1;
}
else{
first = mid;
e = mid - 1;
}
}
cout<<"First Ocurrence of "<<target <<": "<<first<<endl;
cout<<"Last Ocurrence of "<<target <<": "<<last<<endl;
}
```

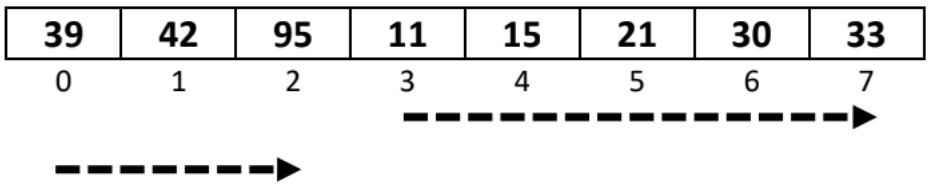
output:

```
Enter the number of elements: 8
Enter elements: 3 4 4 4 5 5 8 9
enter the target element: 4
First Ocurrence of 4: 1
Last Ocurrence of 4: 3
```

Shubham Sharma
2021-22BCSE046

7. Given a rotated/wrapped sorted array of integers and a key value, write an efficient function to determine if the key value exists.

Sample Array:



Key value: 30
Element found at index 6

```
#include <bits/stdc++.h>
using namespace std;
int pivot(vector<int> &nums, int n)
{
	int s = 0;
	int e = n - 1;
	while (s < e){
		int mid = s + (e - s) / 2;
		if (nums[mid] >= nums[0]){
			s = mid + 1;
		}else{
			e = mid;
		}
	}
	return s;
}
int binarySearch(vector<int> &nums, int s, int e, int target){
	while (s <= e){
		int mid = s + (e - s) / 2;
		if (nums[mid] == target){
			return mid;
		}else if (nums[mid] > target){
			e = mid - 1;
		}else{
			s = mid + 1;
		}
	}
	return -1;
}
int main()
{
	int k;
	cout << "Enter the number of elements: ";
	cin >> k;
	vector<int> nums(k);
	cout << "Enter elements: ";
	for (int i = 0; i < k; ++i){
		cin >> nums[i];
	}
	int target;
	cout << "enter the target element: ";
	cin >> target;
```

Shubham Sharma
2021-22BCSE046

```
        int index = -1;
        int piv = pivot(nums, k);
        if (nums[piv] <= target && target <= nums[k - 1]){
                index = binarySearch(nums, piv, k - 1, target);
                cout << "Index of target is : " << index;
        }else{
                index = binarySearch(nums, 0, piv - 1, target);
                cout << "Index of target is : " << index;
        }
}
```

output:

```
Enter the number of elements: 8
Enter elements: 39 42 95 11 15 21 30 33
enter the target element: 30
Index of target is : 6
```

8. Write a function to find all peak elements from an array of integers. A peak element is an element that is greater than or equal to its neighbours. Consider that the input array is wrapped around.
   Sample Array: 100, 20, 15, -2, 23, 90, 67
   Peak elements: 100, 90

```
#include <bits/stdc++.h>
using namespace std;
vector<int> findPeakElements(const vector<int> &nums{
        vector<int> peaks;
        int n = nums.size();
        if (n == 0)
                return peaks;
        if (nums[0] >= nums[1])
                peaks.push_back(nums[0]);
        for (int i = 1; i < n - 1; ++i){
                if (nums[i] >= nums[i - 1] && nums[i] >= nums[i + 1])
                        peaks.push_back(nums[i]);
        }
        if (nums[n - 1] >= nums[n - 2])
                peaks.push_back(nums[n - 1]);
        return peaks;
}
int main(){
        int k;
        cout << "Enter the number of elements: ";
        cin >> k;
        vector<int> nums(k);
        cout << "Enter " << k << " elements: ";
        for (int i = 0; i < k; ++i){
                cin >> nums[i];
        }
        vector<int> peakElements = findPeakElements(nums);
        cout << "Peak elements: ";
        for (int peak : peakElements){
                cout << peak << " ";
        }
}
```

output:

```
Enter the number of elements: 7
Enter 7 elements: 100 20 15 -2 23 90 67
Peak elements: 100 90
```