Experiment 12

AIM-Implementation of Binary Search in real life application

THEORY-Definition: Binary Search is an efficient algorithm for locating a target value within a sorted array. It repeatedly divides the search interval in half, making it faster than linear search.

How It Works:

1. **Initial Setup**: Use two pointers, `low` and `high`, to represent the current search interval.
2. **Iteration**:
    - Calculate the middle index: `mid = (low + high) / 2`.
    - Compare the middle element with the target:
        - If it matches, return the index.
        - If the target is smaller, adjust `high` to `mid - 1`.
        - If the target is larger, adjust `low` to `mid + 1`.
3. **Repeat** until `low` exceeds `high` or the target is found.

Complexity:

- **Time Complexity**: O(logn)O(\log n)O(logn)
- **Space Complexity**: O(1)O(1)O(1) for iterative implementation.

Real-Life Applications of Binary Search

1. **Database Searching**: Quickly find records in a sorted database.
2. **Library Catalogs**: Efficiently locate books in a sorted list.
3. **Autocomplete Systems**: Suggest completions from a sorted word list.
4. **Game Development**: Efficiently find leaderboard scores in a sorted array.
5. **Image Processing**: Locate pixel values in sorted color palettes.

INPUT-

```
#include <stdio.h>

#include <string.h>

#define MAX 100

// Binary Search function

int binarySearch(char contacts[][MAX], int low, int high, char* name) {

    while (low <= high) {
```

```
        int mid = low + (high - low) / 2;

        // Compare middle element with the name
        int result = strcmp(contacts[mid], name)
        // Check if name is present at mid
        if (result == 0)
            return mid;
        // If name is greater, ignore left half
        if (result < 0)
            low = mid + 1;
        // If name is smaller, ignore right half
        else
            high = mid - 1;
    }
    return -1; // Name not found
}
int main() {
    char contacts[][MAX] = {"Alice", "Bob", "Charlie", "David", "Eve", "Frank"};
    int n = sizeof(contacts) / sizeof(contacts[0]);
    char name[MAX];
    printf("Enter the name to search: ");
    scanf("%s", name);
    int result = binarySearch(contacts, 0, n - 1, name);
    if (result != -1)
```

```
        printf("%s found at index %d\n", name, result);
    else
        printf("%s not found in contacts.\n", name);
    return 0;
}
```

Output-

```
/tmp/7QDJ2Yvcqf.o
Enter the name to search: Alice
Alice found at index 0
```

Conclusions-

This is a basic example of how binary search can be applied to real-life applications like contact management.