Experiment No.1

Title:Binary Search Techniques using arrays and recursion.

Applications

1.Binary Search Technique using array.:-

Binary Search in Dictionary and Spell Checker

- **The Problem**

When you type a word (like **"**binary**"**) into a dictionary app or spell checker, the system must quickly check whether that word exists in its word list.

Programm:- #include <stdio.h>

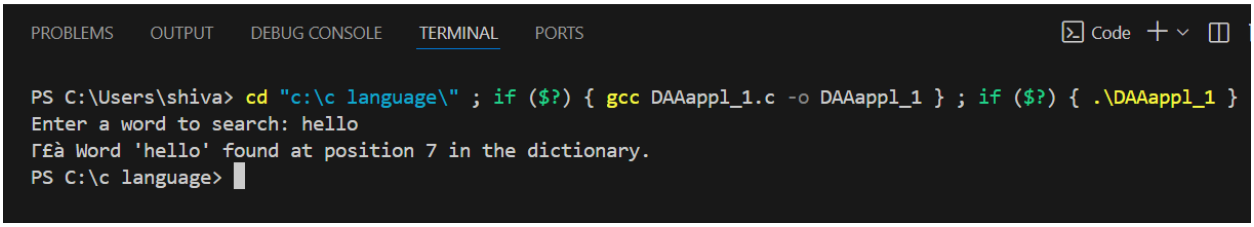
#include <string.h>

```
int binarySearch(char words[][20], int n, char target[]) {
   int low = 0, high = n - 1;
   while (low <= high) {
      int mid = (low + high) / 2;
      int cmp = strcmp(words[mid], target);
      if (cmp == 0)
         return mid;  // Word found
      else if (cmp > 0)
         high = mid - 1;  // Search left half
      else
         low = mid + 1;   // Search right half
   }
   return -1;  // Word not found
}
int main() {
```

```
    // Sorted dictionary (alphabetical order)
    char words[][20] = {
        "apple", "ball", "binary", "book", "cat",
        "code", "data", "hello", "world"
    };
    int n = sizeof(words) / sizeof(words[0]);
    char target[20];
    printf("Enter a word to search: ");
    scanf("%s", target);
    int result = binarySearch(words, n, target);
    if (result != -1)
        printf("✅ Word '%s' found at position %d in the dictionary.\n", target, result);
    else
        printf("❌ Word '%s' not found! (Maybe check spelling?)\n", target);

    return 0;
}
```

Output:-



2.Binary Search Technique using Recursion:-

Problem: Towers of Hanoi

You have 3 pegs and n disks of different sizes stacked on one peg (source) in increasing size order.

The goal is to move all disks to another peg (destination), using one extra peg (auxiliary), following two rules:

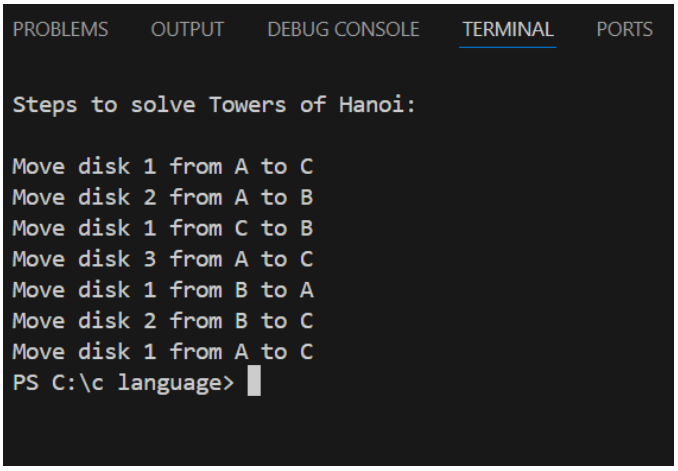
1. You can move only one disk at a time.
2. You cannot place a larger disk on top of a smaller disk.

Programm:-

```
#include <stdio.h>
// Recursive function to solve Towers of Hanoi
void towerOfHanoi(int n, char source, char auxiliary, char destination) {
   if (n == 1) {
      printf("Move disk 1 from %c to %c\n", source, destination);
      return;
   }
   // Move n-1 disks from source to auxiliary
   towerOfHanoi(n - 1, source, destination, auxiliary);
   // Move the remaining disk from source to destination
   printf("Move disk %d from %c to %c\n", n, source, destination);
   // Move the n-1 disks from auxiliary to destination
   towerOfHanoi(n - 1, auxiliary, source, destination);
}
int main() {
   int n;
   printf("Enter the number of disks: ");
   scanf("%d", &n);
   printf("\nSteps to solve Towers of Hanoi:\n\n");
   towerOfHanoi(n, 'A', 'B', 'C'); // A = source, B = auxiliary, C = destination
   return 0;
```

}

Output:-



Real-World Applications of Binary Search

1. Searching in Databases
    - Quickly find a record in a sorted database, like employee IDs, student roll numbers, or product codes.
    - Recursive binary search is ideal for hierarchical or indexed data.
2. Dictionary and Word Lookup
    - Finding words in digital dictionaries or spell-check systems efficiently.
    - Each lookup can use a recursive binary search on sorted word arrays.
3. E-Commerce Product Search
    - Searching for a product by price, ID, or rating in a sorted catalog.
    - Helps quickly locate items without scanning the entire array.
4. Computer Graphics
    - Finding intersections in sorted arrays of pixels, layers, or geometric coordinates.
    - Recursive search speeds up collision detection or rendering.
5. Network Routing

- Searching sorted routing tables to determine the next hop efficiently.

6. Version Control Systems
    - Using binary search to find a specific commit in a sorted history for debugging or rollback.
7. Medical Data Analysis
    - Searching for a patient record in sorted medical datasets by ID or date.
8. Real-Time Systems
    - Recursive binary search is used in embedded systems to quickly locate thresholds, sensor ranges, or calibration points in sorted arrays.