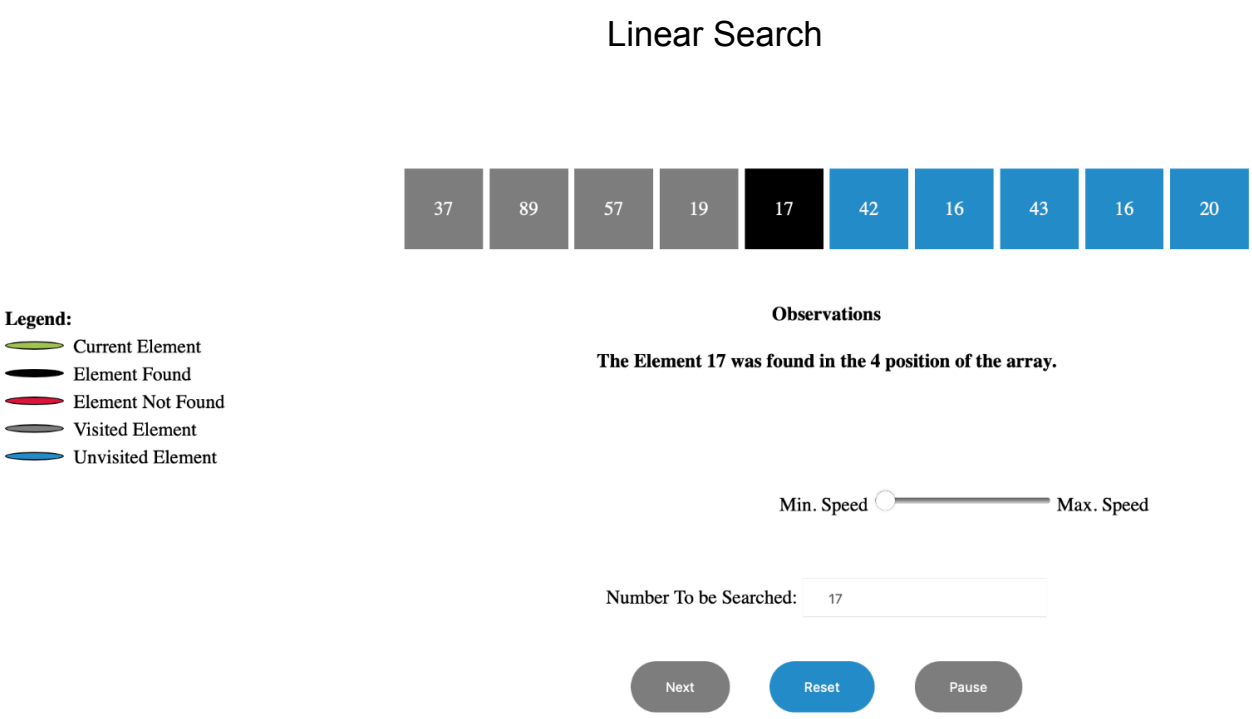
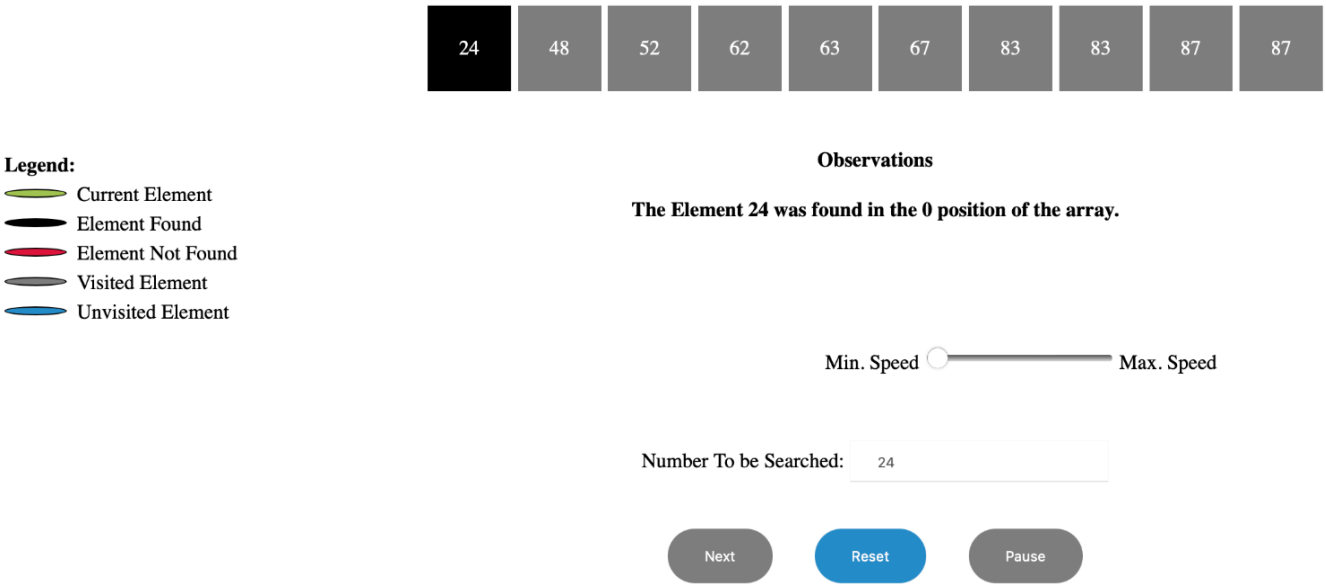
15B17CI371 – Data Structures Lab

ODD 2024
Week 4-LAB B
Practice Lab

Virtual Lab



Binary Search



1.

```
#include <iostream>
using namespace std;
#define MAX_SIZE 100
void countFrequencies(int arr[], int size) {
    int uniqueElements[MAX_SIZE];
    int frequencies[MAX_SIZE];
    int uniqueCount = 0;
    for (int i = 0; i < MAX_SIZE; ++i) {
        uniqueElements[i] = -1;
        frequencies[i] = 0;
    }
    for (int i = 0; i < size; ++i) {
        int element = arr[i];
        bool found = false;

        for (int j = 0; j < uniqueCount; ++j) {
            if (uniqueElements[j] == element) {
                frequencies[j]++;
                found = true;
```

```
            break;
        }
    }
    if (!found) {
        uniqueElements[uniqueCount] = element;
        frequencies[uniqueCount] = 1;
        uniqueCount++;
    }
  }
  cout << "Unique: {";
  for (int i = 0; i < uniqueCount; ++i) {
    cout << uniqueElements[i];
    if (i < uniqueCount - 1) cout << ", ";
  }
  cout << "}" << endl;
  cout << "Frequency: {";
  for (int i = 0; i < uniqueCount; ++i) {
    cout << frequencies[i];
    if (i < uniqueCount - 1) cout << ", ";
  }
  cout << "}" << endl;
}
int main() {
  int array[] = {9, 12, 3, 31, 3, 19, 9, 3};
  int size = sizeof(array) / sizeof(array[0]);
  countFrequencies(array, size);
  return 0;
}
```

```
Unique: {9, 12, 3, 31, 19}
Frequency: {2, 1, 3, 1, 1}
```

2.

```
#include <iostream>
#include <cmath>
using namespace std;
int jumpSearch(int arr[], int size, int key) {
  int step = sqrt(size);
  int prev = 0;
  while (arr[min(step, size) - 1] < key) {
    prev = step;
    step += sqrt(size);
    if (prev >= size) return -1;
  }
```

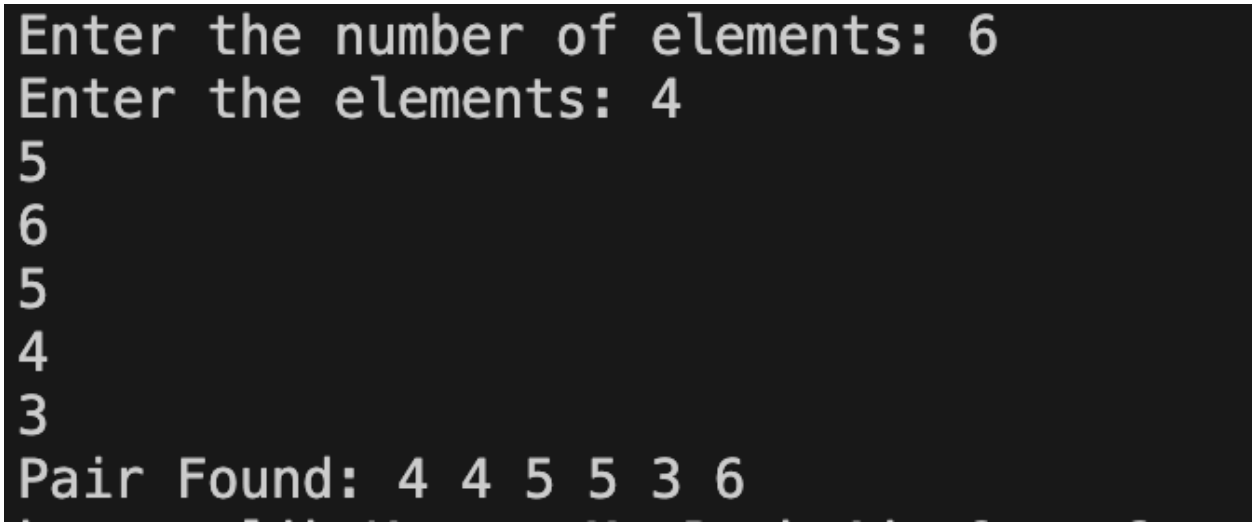
```
    while (arr[prev] < key) {
        prev++;
        if (prev == min(step, size)) return -1;
    }
    if (arr[prev] == key) return prev;
    return -1;
}
int main() {
    int size;
    cout << "Enter the number of elements: ";
    cin >> size;
    if (size <= 0) {
        cout << "Array size must be positive." << endl;
        return 1;
    }
    int* array = new int[size];
    cout << "Enter the elements (sorted): ";
    for (int i = 0; i < size; ++i) {
        cin >> array[i];
    }
    int key;
    cout << "Enter the key to search: ";
    cin >> key;
    int index = jumpSearch(array, size, key);
    if (index != -1) {
        cout << "Element found at index " << index << endl;
    } else {
        cout << "Element not found" << endl;
    }
    delete[] array;
    return 0;
}
```

```
Enter the number of elements: 5
Enter the elements (sorted): 2
4
6
8
10
Enter the key to search: 4
Element found at index 1
```

3.

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

const int MAX_SIZE = 100;

void countFrequency(int arr[], int n, int unique[], int freq[], int& uniqueCount) {
   uniqueCount = 0;
   for (int i = 0; i < n; ++i) {
      bool found = false;
      for (int j = 0; j < uniqueCount; ++j) {
         if (arr[i] == unique[j]) {
            freq[j]++;
            found = true;
            break;
         }
      }
      if (!found) {
         unique[uniqueCount] = arr[i];
         freq[uniqueCount] = 1;
         uniqueCount++;
      }
   }
}

void sortByFrequency(int unique[], int freq[], int n) {
   for (int i = 0; i < n - 1; ++i) {
      for (int j = i + 1; j < n; ++j) {
         if (freq[i] < freq[j] || (freq[i] == freq[j] && unique[i] > unique[j])) {
            swap(freq[i], freq[j]);
            swap(unique[i], unique[j]);
         }
      }
   }
}

void sortArrayByFrequency(int input[], int size) {
   int freq[MAX_SIZE];
   int unique[MAX_SIZE];
   int uniqueCount;
```

```
    countFrequency(input, size, unique, freq, uniqueCount);
    sortByFrequency(unique, freq, uniqueCount);

    cout << "Pair Found: ";
    for (int i = 0; i < uniqueCount; ++i) {
        for (int j = 0; j < freq[i]; ++j) {
            cout << unique[i] << " ";
        }
    }
    cout << endl;
}

int main() {
    int size;
    cout << "Enter the number of elements: ";
    cin >> size;

    if (size <= 0 || size > MAX_SIZE) {
        cout << "Invalid size. Size must be positive and less than or equal to " << MAX_SIZE << endl;
        return 1;
    }

    int array[MAX_SIZE];
    cout << "Enter the elements: ";
    for (int i = 0; i < size; ++i) {
        cin >> array[i];
    }

    sortArrayByFrequency(array, size);

    return 0;
}
```



4.

```
#include <iostream>

using namespace std;

#define MAX_SIZE 100

int absolute(int value) {
   return (value < 0) ? -value : value;
}

void computeAndSortDifferences(const int arr[], int size, int out[], int& outSize) {
   if (size < 2) {
      outSize = 0;
      return;
   }

   int differences[MAX_SIZE];
   int diffCount = 0;

   for (int i = 1; i < size; ++i) {
      int diff = arr[i] - arr[i - 1];
      differences[diffCount++] = absolute(diff);
   }

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

```
        for (int j = i + 1; j < diffCount; ++j) {
            if (differences[j] > differences[i]) {
                int temp = differences[i];
                differences[i] = differences[j];
                differences[j] = temp;
            }
        }
    }

    for (int i = 0; i < diffCount; ++i) {
        out[i] = differences[i];
    }

    outSize = diffCount;
}

void printArray(const int arr[], int size) {
    cout << "{";
    for (int i = 0; i < size; ++i) {
        cout << arr[i];
        if (i < size - 1) cout << ", ";
    }
    cout << "}" << endl;
}

int main() {
    int size;
    cout << "Enter the number of elements: ";
    cin >> size;

    if (size < 2 || size > MAX_SIZE) {
        cout << "Invalid size. Size must be between 2 and " << MAX_SIZE << "." << endl;
        return 1;
    }

    int array[MAX_SIZE];
    cout << "Enter the elements: ";
    for (int i = 0; i < size; ++i) {
        cin >> array[i];
    }

    int result[MAX_SIZE];
```

```
    int resultSize;

    computeAndSortDifferences(array, size, result, resultSize);

    cout << "Sorted differences in descending order: ";
    printArray(result, resultSize);

    return 0;
}
```