INTERPOLATION SEARCH CODE IMPLEMENTATION

ARRAY

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

int interpolationSearch(int arr[], int n, int x) {
   int low = 0, high = (n - 1);

   while (low <= high && x >= arr[low] && x <= arr[high]) {
      // Calculate the position using interpolation formula
      int pos = low + ((x - arr[low]) * (high - low) / (arr[high] - arr[low]));

      // Check if the element is present at the position
      if (arr[pos] == x) {
         return pos;
      }

      // If x is greater, x is in the right subarray
      if (arr[pos] < x) {
         low = pos + 1;
      }
      // If x is smaller, x is in the left subarray
      else {
         high = pos - 1;
      }
   }
```

```
    // Element not found
    return -1;
}

int main() {
    int arr[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 70; // Element to be searched

    int index = interpolationSearch(arr, n, x);

    if (index != -1) {
        cout << "Element found at index " << index << endl;
    } else {
        cout << "Element not found in array" << endl;
    }

    return 0;
}
```

OUTPUT

```
C:\Users\studentuser\Desktop\DSA\Lectures\WEEK 02\interpolation search array.exe
Element found at index 6

--------------------------------
Process exited after 0.07702 seconds with return value 0
Press any key to continue . . .
```

LINKED LIST

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

// Definition for singly-linked list node.
class ListNode {
public:
    int val;
    ListNode* next;

    ListNode(int x) : val(x), next(nullptr) {}
};

// Definition for LinkedList class.
class LinkedList {
public:
    ListNode* head;

    LinkedList() : head(nullptr) {}

    // Function to add a new node to the linked list.
    void push(int new_data) {
        ListNode* new_node = new ListNode(new_data);
        new_node->next = head;
        head = new_node;
```

```
}

// Function to find the length of the linked list.
int length() {
    int len = 0;
    ListNode* temp = head;
    while (temp != nullptr) {
        len++;
        temp = temp->next;
    }
    return len;
}

// Function to find a node by index.
ListNode* getNodeAt(int index) {
    ListNode* temp = head;
    int count = 0;
    while (temp != nullptr && count < index) {
        temp = temp->next;
        count++;
    }
    return temp;
}

// Function to perform interpolation search on the linked list.
ListNode* interpolationSearch(int target) {
    int len = length();
```

```
    ListNode* start = head;
    ListNode* end = nullptr;

    while (start != end && start != nullptr) {
        // Find the index positions for the interpolation
        int lowIndex = 0;
        int highIndex = len - 1;

        // Find the nodes at the low and high indices
        ListNode* lowNode = start;
        ListNode* highNode = getNodeAt(highIndex);

        if (lowNode == nullptr || highNode == nullptr) return nullptr;

        // Estimate the position using interpolation formula
        int pos = lowIndex + ((target - lowNode->val) * (highIndex - lowIndex)) / (highNode->val -
lowNode->val);

        // Find the node at the estimated position
        ListNode* midNode = getNodeAt(pos);

        if (midNode == nullptr) return nullptr;

        // Check if the midNode is the target
        if (midNode->val == target) {
            return midNode;
        }
```

```
        // If target is smaller, search the left half
        else if (midNode->val > target) {
            end = midNode;
        }
        // If target is larger, search the right half
        else {
            start = midNode->next;
            lowIndex = pos + 1; // Adjust low index for the next iteration
        }
    }

    // If we reach here, the target is not present in the list
    return nullptr;
  }

  // Function to print the linked list (for debugging purposes)
  void printList() {
    ListNode* temp = head;
    while (temp != nullptr) {
        cout << temp->val << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
  }
};

int main() {
```

```
    // Create a LinkedList object and add elements to it.
    LinkedList list;

    // Insert elements (manually adding in sorted order for simplicity)
    list.push(9);
    list.push(7);
    list.push(5);
    list.push(3);
    list.push(1);

    // Print the list (optional)
    cout << "Linked List: ";
    list.printList();

    int target = 5;

    // Perform interpolation search
    ListNode* result = list.interpolationSearch(target);
    if (result != nullptr) {
        cout << "Element found with value " << result->val << endl;
    } else {
        cout << "Element not found" << endl;
    }

    return 0;
}
```

OUTPUT

```
C:\Users\studentuser\Desktop\DSA\Lectures\WEEK 02\interpolation search linked list.exe
Linked List: 1 -> 3 -> 5 -> 7 -> 9 -> NULL
Element found with value 9

--------------------------------
Process exited after 0.07435 seconds with return value 0
Press any key to continue . . .
```