1. **Problem Statement**

Create a recursive function to search through an array using a linear search and returns number of recursive calls were made.

1. **Requirements**
   1. **Assumptions**
      1. We deal with integer number throughout the code
      2. User enters integer value for key
   2. **Specifications**
      1. Initialize variable for size
      2. Load values in an array starting with 0
      3. Get a key from user
      4. Call the function *recursiveLinearSearch* passing in the size, key, array, and status of method
         1. This function should have recursive method that goes through each array index to find the key given by user
         2. If key is not found, change method’s status to false
      5. Returns the number of times recursive calls were made
2. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Key | Passed in to the function *recursiveLinearSearch* along with array, size, and methodStatus |  |
|  | Recursive function goes through each index of array and finds the index where the key is located | The number of times recursive calls were made |
|  | Checks if key is not found | Not found message and number of calls |

1. **Test Strategy**
   1. Valid Data
   2. Invalid Data
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User inputs a positive number inside the range |  |  |  |  |
| Invalid | 1 | User inputs a negative number |  |  |  |  |
| Valid | 2 | User inputs out of range number |  |  |  |  |

1. **Initial Algorithm**
   1. In main function
      1. Initialize variable size to 5000
      2. Initialize array *array[]* and using for loop fill up the array counting from 0 at 0th index and increment by 1
         1. Index 0 has value 0, index 1 has value 1, etc…
      3. Ask user, “What number would you like to search for: ” and assign the value to an integer variable *key*
      4. Pass in the variables size, key, array, and a Boolean variable *methodStatus*
         1. *methodStatus* will be changed to false if key not found and vice versa
      5. Print out the number of function calls returned from calling the recursive function
   2. In *recursiveLinearSearch* function:
      1. Check if key equals the size - 1 index of array
         1. If it equals that, return 1 and change *methodStatus* to true
      2. Else
         1. Set a variable position to the result of calling recursive function again, where size – 1 is passed in along with the array, key variable, and methodStatus variable
         2. Return position + 1
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User inputs a positive number inside the range | 699 | “Found after 1 recursion calls” |  |  |
| Invalid | 1 | User inputs a negative number | -500 and change size to 700 to not have stack overflow problem | “Key Not Found” |  |  |
| Invalid | 2 | User inputs positive out of range number | 6000 and change size to 700 to not have stack overflow problem | “Key Not Found” |  |  |

1. **Code**

#include <iostream>

#include <string>

using namespace std;

int recursiveLinearSearch(int array[], int key, int size, bool &methodStatus);

int main()

{

const int size = 700;

int array[size];

int key;

bool methodStatus;

for (int i = 0; i < size; i++)

{

array[i] = i;

}

do {

cout << "What number would you like to search for: " << endl;

cin >> key;

if (key < 0)

{

cout << "Your input is negative. Key not found. Please try again" << endl;

}

else if (key >= size)

{

cout << "Your input is out of range. Key not found. Please try again with numbers between 0 and 4999" << endl;

}

} while (key < 0 || key >= size);

cout << "Found after " << recursiveLinearSearch(array, key, size, methodStatus) << " recursion calls" << endl;

system("pause");

return 0;

}

int recursiveLinearSearch(int array[], int key, int size, bool &methodStatus)

{

int pos;

if (key == array[size - 1])

{

methodStatus = true;

return 1;

}

else {

pos = recursiveLinearSearch(array, key, size - 1, methodStatus);

return pos + 1;

}

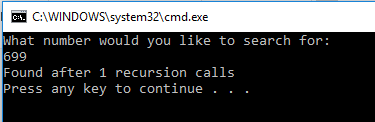
}

1. **Updated Algorithm**
   1. In main function
      1. Initialize variable size to 5000
      2. Initialize Boolean variable methodStatus and int variable key
      3. Initialize array *array[]* and set its size using variable size and using for loop fill up the array counting from 0 at 0th index and increment by 1
         1. Index 0 has value 0, index 1 has value 1, etc…
      4. Ask user, “What number would you like to search for: ” and assign the value to an integer variable *key*
         1. Using a do while loop, keep asking this until the user enters a number between 0 and 4999. Print out the error if the user input is either negative or more than 4999.
      5. Pass in the variables size, key, array, and a Boolean variable *methodStatus*
         1. *methodStatus* will be changed to false if key not found and vice versa
      6. Print out the number of function calls returned from calling the recursive function
   2. In *recursiveLinearSearch* function:
      1. Check if key equals the size - 1 index of array
         1. If it equals that, return 1 and change *methodStatus* to true
      2. Else
         1. Initialize variable for position, which is a counter variable of how many recursion calls will be made
         2. Set a variable position to the result of calling recursive function again, where size – 1 is passed in along with the array, key variable, and methodStatus variable
         3. Return position + 1
2. **Test Plan Version 3**

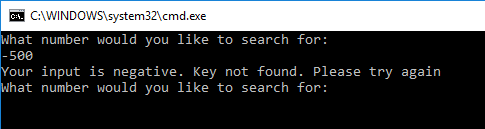
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User inputs a positive number inside the range | 699 | “Found after 1 recursion calls” | “Found after 1 recursion calls” | Pass |
| Invalid | 1 | User inputs a negative number | -500 and change size to 700 to not have stack overflow problem | “Key Not Found” | “Key Not Found” | Pass |
| Invalid | 2 | User inputs positive out of range number | 6000 and change size to 700 to not have stack overflow problem | “Key Not Found” | “Key Not Found” | Pass |

1. **Screenshots**

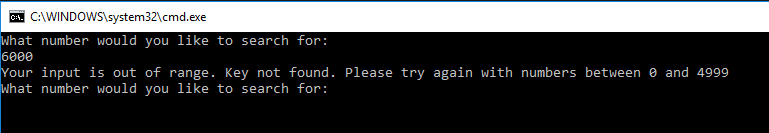
Valid Test Case 1:



Invalid Test Case 1:



Invalid Test Case 2:



1. **Status**

Program runs perfectly with assumptions and tests in mind.