Bangladesh University of Business and Technology

Department of Computer Science and Engineering

CSE 232: Data Structures Lab

Solution to Lab 04 Tasks

Task 1

Write functions for performing Binary Search using -

- Iterative Method
- Recursive Method

The program will first take the number of components in the array as input. Followed by the actual components of the array. Then finally the “value” that we are searching for.
It should return the index of the “value” in the array.
Note: It should return “-1” if the “value” cannot be found in the array.

Sample Input: 7 1 5 9 13 18 22 35 13	**Sample Output:** 3
Sample Input: 7 1 5 9 13 18 22 35 19	**Sample Output:** -1

Using Iterative Method:

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

int binarySearch(int arr[], int left, int right, int x)
{
  while (left <= right)
  {
    int mid = (left + right) / 2;
    if (arr[mid] == x)
      return mid;
    if (arr[mid] < x)
      left = mid + 1;
    else
      right = mid - 1;
  }
  return -1;
}

int main()
{
  int n, value;
  cin >> n;
  int numberArray[n];
  for (int i = 0; i < n; i++)
    cin >> numberArray[i];
  cin >> value;
  int result = binarySearch(numberArray, 0, n - 1, value);
  cout << result << endl;
}
```

Using Recursive Method:

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

int binarySearch(int arr[], int left, int right, int x)
{
  if (left <= right)
  {
    int mid = (left + right) / 2;

    // If found at mid, then return it
    if (arr[mid] == x)
      return mid;

    // Search the left half
    if (arr[mid] > x)
      return binarySearch(arr, left, mid - 1, x);

    // Search the right half
    else
      return binarySearch(arr, mid + 1, right, x);
  }
  return -1;
}

int main()
{
  int n, value;
  cin >> n;
  int numberArray[n];
  for (int i = 0; i < n; i++)
    cin >> numberArray[i];
  cin >> value;
  int result = binarySearch(numberArray, 0, n - 1, value);
  cout << result << endl;
}
```

Task 2

Modify any one of the Binary Search functions (Iterative or Recursive) from task 1 so that if the “value” cannot be found, it will return the index of the closest number smaller than the “value”.

Sample Input:	**Sample Output:**

7
1 5 9 13 18 22 35
13
	3

Sample Input: 7 1 5 9 13 18 22 35 19	**Sample Output:** 4

Sample Solution:

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

int binarySearch(int arr[], int left, int right, int x)
{
  while (left <= right)
  {
    int mid = (left + right) / 2;

    if (arr[mid] == x)
      return mid;

    if (arr[mid] < x)
      left = mid + 1;

    else
      right = mid - 1;
  }
  return right;
}

int main()
{
  int n, value;
  cin >> n;
  int numberArray[n];
  for (int i = 0; i < n; i++)
    cin >> numberArray[i];
  cin >> value;
  int result = binarySearch(numberArray, 0, n - 1, value);
  cout << result << endl;
}
```

Task 3

Rayne is a hard-working programmer. However, to give her best, she needs to drink coffee from time to time. Suppose there are n cafes in her town and the ith cafe sells a cup of coffee at pi taka. If Rayne has m amount of money, your job is to find how many cafes she can buy a cup of coffee from. Hint: Use the modified binary search function from task 2. You can also use upper_bound() or lower_bound().] The program will take the number of cafes as input first. Followed by the prices of coffee in each of them. And finally, the amount of money Rayne has. It should return the number of cafes from where she can buy coffee.

Sample Input:	**Sample Output:**

5
25 60 75 90 120
30
	1

Sample Input: 7 25 60 75 90 120 250 333 275	**Sample Output:** 6

Sample Solution:

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

int binarySearch(int arr[], int left, int right, int x)
{
  while (left <= right)
  {
    int mid = (left + right) / 2;
    if (arr[mid] == x)
      return mid;
    if (arr[mid] < x)
      left = mid + 1;
    else
      right = mid - 1;
  }
  return right;
}

int main(void)
{
  int n, value;
  cin >> n;
  int arr[n];
  for (int i = 0; i < n; i++)
    cin >> arr[i];
  cin >> value;
  int result = binarySearch(arr, 0, n - 1, value) + 1;
  cout << result << endl;
}
```

Task 4

[Bonus] Charlie from "Charlie and the Chocolate Factory" is looking for the golden ticket. He got a tip-off from a random beggar that the chocolate bar with code **x** has the golden ticket. Charlie knows that there are **n** number of stores in his town and that the i^{th} store has A_i chocolate bars. He also knows that the code for the chocolate bars on the i^{th} store starts from $A_{i-1} + 1$ and increases by 1 increment. The code for the chocolate bars in the 0^{th} store always starts from 1.

For example, if the 0^{th} store has 125 chocolate bars, then the codes for those bars range from 1 to 125. Then, if the 1^{st} store has 300 chocolate bars, the codes for those bars range from 126 to 426.

Your job is to tell Charlie from which store he should buy the chocolate bar. You are given the number of stores in his town, followed by the number of chocolate bars in each store. And finally, the code for the chocolate bar which has the golden ticket.

Sample Input:	**Sample Output:**

5
20 45 10 17 33
77
	3

Sample Input:	**Sample Output:**

3
120 250 333
447
	2

Explanation (Sample Input 1):
Code for chocolate bars in **Store 0**: 1 to 20
Code for chocolate bars in **Store 1**: 21 to 65
Code for chocolate bars in **Store 2**: 66 to 75
Code for chocolate bars in **Store 3**: 76 to 92
Code for chocolate bars in **Store 4**: 93 to 125
So, Store 3 has the chocolate bar with the code 77

Sample Solution:

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

int binarySearch(int arr[], int left, int right, int x)
{
    while (left <= right)
    {
        int mid = (left + right) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return right;
}

int main()
{
    int n, x;
    cin >> n;
    int chocolates, startingCode[n + 1];
    memset(startingCode, 0, n + 1);
    startingCode[0] = 1;
    for (int i = 1; i < n + 1; i++)
    {
        cin >> chocolates;
        startingCode[i] = startingCode[i - 1] + chocolates;
    }
    cin >> x;
    if (startingCode[n] <= x)
        cout << "Sorry Charlie. The chocolate bar is not in this town :(" << endl;
    else
    {
        int storeNo = binarySearch(startingCode, 0, n - 1, x);
        cout << storeNo << endl;
    }
}
```