**Edwin Ario Abdiwijaya**

[**edwin.abdiwijaya@binus.ac.id**](mailto:edwin.abdiwijaya@binus.ac.id)

**Answer sheet for assignment 4**

Answer for question 1 :

Because if we use (beg + end) / 2, the code may be overflow. For example, a type integer can hold only 10000, if beg = 9990 and end = 9995, when beg + end, the result 19985 where the result exceed the limit of that type integer. That’s why we should use beg + (end – beg) / 2.

Answer for question 2 :

// Edwin Ario Abdiwijaya

// edwin.abdiwijaya@binus.ac.id

// Answer for question number 2

#include <stdio.h>

int ternarySearch(int l, int r, int idx, int number[])

{

    while (r >= l)

    {

        int lmid = l + (r - l) / 3;

        int hmid = r - (r - l) / 3;

        if (number[lmid] == idx)

        {

            return lmid;

        }

        else if (number[hmid] == idx)

        {

            return hmid;

        }

        else if (idx > number[hmid])

        {

            l = hmid + 1;

        }

        else if (idx < number[lmid])

        {

            r = lmid - 1;

        }

        else

        {

            l = lmid + 1;

            r = hmid - 1;

        }

    }

    return -1;

}

int main()

{

    int N, l, r, p, idx, number[10000], i, j, M;

    printf("How much number you want to input to array?\n");

    printf(">> ");

    scanf("%d", &N);

    getchar();

    for (i = 0; i < N; i++)

    {

        printf("Input the number %d : ", i + 1);

        scanf("%d", &number[i]);

        getchar();

    }

    l = 0;

    r = N - 1;

    printf("\n");

    printf("How many times you want to search the number?\n");

    printf(">> ");

    scanf("%d", &M);

    getchar();

    for (j = 0; j < M; j++)

    {

        printf("Enter number to be find : ");

        scanf("%d", &idx);

        getchar();

        p = ternarySearch(l, r, idx, number);

        if (p >= 0)

        {

            printf("%d is found at index %d\n", idx, p);

        }

        else

        {

            printf("%d is not found in the array\n", idx);

        }

        printf("\n");

    }

    return 0;

}