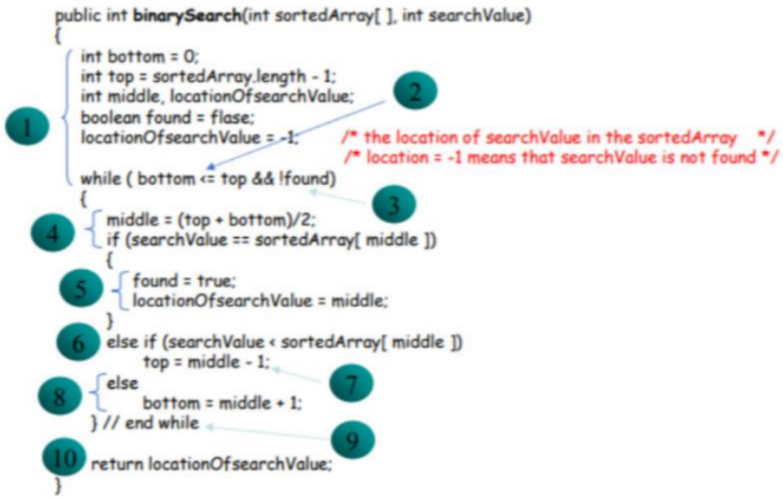
LAB 05: Binary Search and Bubble Sort Program Testing

TASK 1

Recall the logic of the Binary Search Algorithm, Code, and Test it using Any Method. /*Attach a printout of code & Test Execution Summary here*/

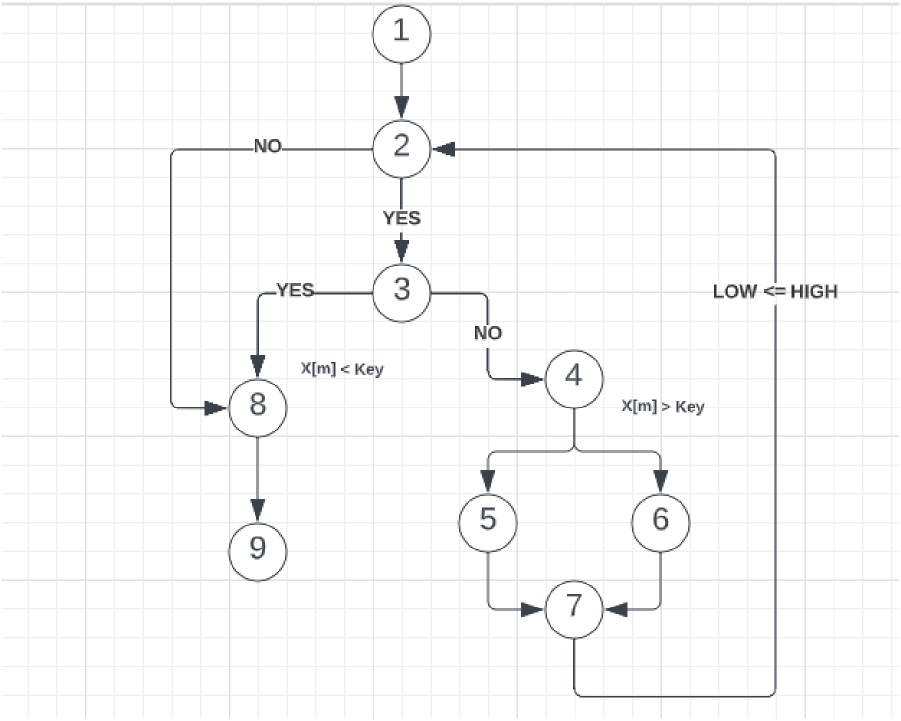


```
public int binarySearch(int sortedArray[ ], int searchValue)
{
    int bottom = 0;
    int top = sortedArray.length - 1;
    int middle, locationOfsearchValue;
    boolean found = flase;
    locationOfsearchValue = -1;        /* the location of searchValue in the sortedArray    */
                                       /* location = -1 means that searchValue is not found */
    while ( bottom <= top && !found)
    {
      middle = (top + bottom)/2;
      if (searchValue == sortedArray[ middle ])
      {
        found = true;
        locationOfsearchValue = middle;
      }
      else if (searchValue < sortedArray[ middle ])
            top = middle - 1;
      else
            bottom = middle + 1;
    } // end while
    return locationOfsearchValue;
}
```

CODE

```
#include<stdio.h>
int binary_search(int x[],int low,int high,int key)
{
   int m;
  while(low<=high)
  {
            m=(low+high)/2;
            if(x[m]==key)
            return m;
            if(x[m]<key)
            low=m+1;
            else
            high=m-1;
  }
  return -1;
}

int main()
{
  int a[20],key,i,n,succ;
  printf("Enter the n value");
  scanf("%d",&n);
  if(n>0)
  {
                printf("enter the elements in ascending order\n");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);

          printf("enter the key element to be searched\n");
          scanf("%d",&key);
    succ=binary_search(a,0,n-1,key);
             if(succ>=0)
      printf("Element found in position = %d\n",succ+1);
             else
       printf("Element not found \n");
  }
  else
    printf("Number of element should be greater than zero\n");
             return 0;
}
```

CONTROL FLOW GRAPH



CYCLOMATIC COMPLEXITY - V(G)

Since V(G) = E - N + 2

Where;
E = Number of edges = 11
N = Number of nodes = 9

Therefore;
V(G) = 11 - 9 + 2
= 4

NUMBER OF INDEPENDENT PATHS

Following are the independent paths:

P1: 1-2-3-8-9
P2: 1-2-3-4-5-7-2
P3: 1-2-3-4-6-7-2
P4: 1-2-3-4-6-7-2-8-9

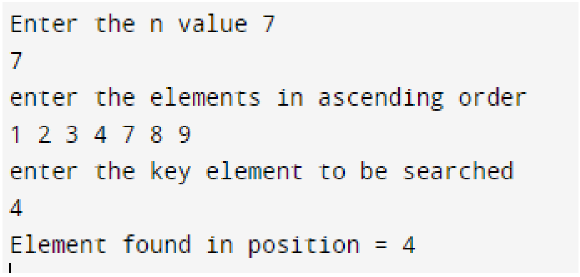
TEST DESCRIPTION

Project Name	Binary Search Algorithm
Test Case ID	Binary_Search_TESE37
Test Title	Testing the program output
Test Priority	Medium
Module Name	Test_Algorithm
Test Data	Enter the 2 Integer Values
Designed By	Sufiyan Irfan
Designed Date	09-07-2022
Executed By	Sufiyan Irfan
Execution Date	09-07-2022
Description of Test	Verify the output of the program

TEST CASES

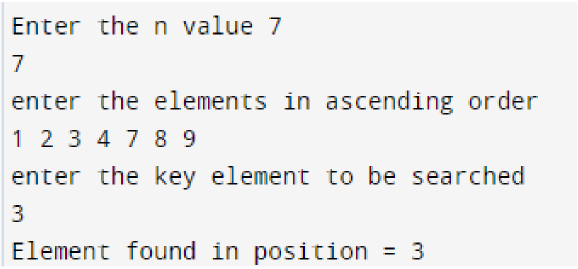
Case ID	Description	Input Data		Expected Output (location)	Actual Output (location)	Status
		X[ ]	Key			
1	Binary search Algorithm P1(1-2-3-8-9)	[1,2,3,4,7,8,9]	4	Element found in position = 4	Element found in position = 4	PASS
2	Binary search Algorithm P2(1-2-3-4-5-7-2)	[1,2,3,4,7,8,9]	3	Element found in position = 3	Element found in position = 3	PASS
3	Binary search Algorithm P3(1-2-3-4-6-7-2)	[1,2,3,4,7,8,9]	8	Element found in position = 6	Element found in position = 6	PASS
4	Binary search Algorithm P4(1-2-3-4-6-7-2-8-9)	[1,2,3,4,7,8,9]	15	Element not found	Element not found	PASS

Test Case 1



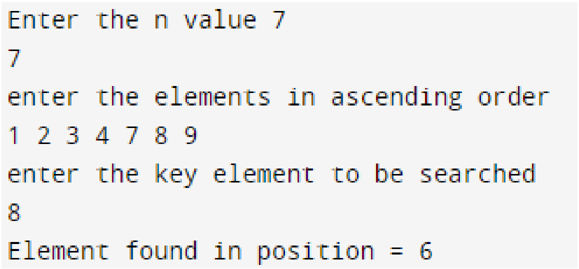
```
Enter the n value 7
7
enter the elements in ascending order
1 2 3 4 7 8 9
enter the key element to be searched
4
Element found in position = 4
```

Test Case 2



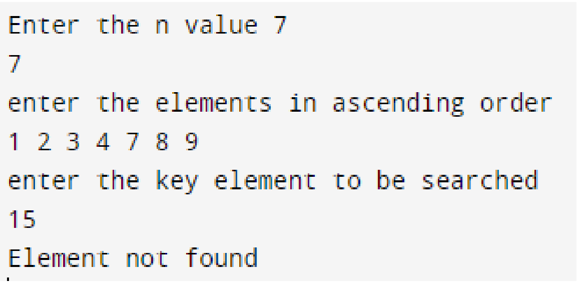
```
Enter the n value 7
7
enter the elements in ascending order
1 2 3 4 7 8 9
enter the key element to be searched
3
Element found in position = 3
```

Test Case 3



```
Enter the n value 7
7
enter the elements in ascending order
1 2 3 4 7 8 9
enter the key element to be searched
8
Element found in position = 6
```

Test Case 4



```
Enter the n value 7
7
enter the elements in ascending order
1 2 3 4 7 8 9
enter the key element to be searched
15
Element not found
```

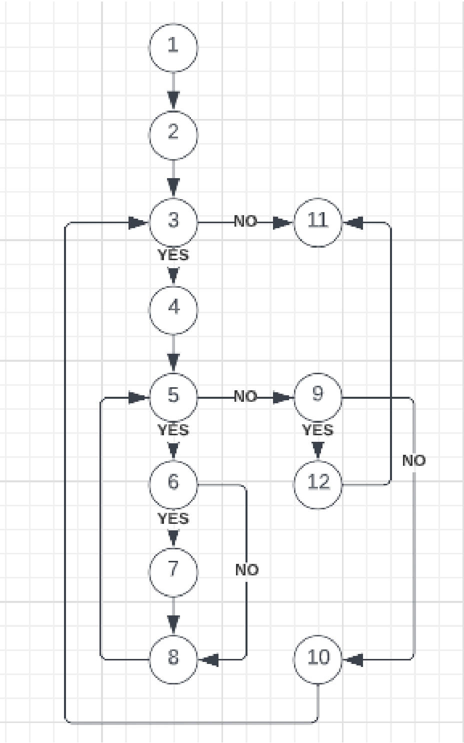
TASK 2

Recall the logic of the bubble sort algorithm, and Test it using Any Method using a template. /* Attach printout of code & Test Execution Summary here*/

CODE

```
#include<iostream>

using namespace std;

int main ()

{

  int i, j,temp=0;

  bool flag=true;

  int a[5] = {3,1,2,5,4};

  cout <<"Input list ...\n";

  for(i = 0; i<5; i++) {

    cout <<a[i]<<"\t";

  }

cout<<endl;

for(i = 0; i<5; i++) {

  for(j = i+1; j<5; j++)

  {

    if(a[j] < a[i]) {

      temp = a[i];

      a[i] = a[j];

      a[j] = temp;

    }

  }

  if(!flag) break;

}

cout <<"Sorted Element List ...\n";

for(i = 0; i<5; i++) {
  cout <<a[i]<<"\t";

}


return 0;

}
```

CONTROL FLOW GRAPH



CYCLOMATIC COMPLEXITY - V(G)

Since V(G) = E - N + 2

Where;
E = Number of edges = 15
N = Number of nodes = 12

Therefore;
V(G) = 15 - 12 + 2
= 5

NUMBER OF INDEPENDENT PATHS

Following are the independent paths:

P1: 1-2-3-4-5-6-7-8-9-10-11
P2: 1-2-3-11
P3: 1-2-3-4-5-9-10-11
P4: 1-2-3-4-5-9-12-11
P5: 1-2-3-4-5-6-8-9-12-11

TEST DESCRIPTION

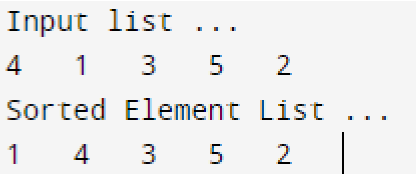
Project Name	Bubble Sort Algorithm
Test Case ID	Bubble_Sort_TESE37
Test Title	Testing the program output
Test Priority	Medium
Module Name	Test_Algorithm
Test Data	Enter the 2 Integer Values
Designed By	Sufiyan Irfan
Designed Date	09-07-2022
Executed By	Sufiyan Irfan
Execution Date	09-07-2022
Description of Test	Verify the output of the program

TEST CASES

Case ID	**Description**	**Input Data**	**Expected Output (location)**	**Actual Output (location)**	**Status**
		int [ ] array			
1	Enter array and verify bubble sort algorithm P1(1,2,3,4,5,6,7,8,9,10,11)	[4,1,3,5,2]	[1,2,3,4,5]	[1,2,3,4,5]	PASS
2	Enter array and verify bubble sort algorithm P2(1,2,3,4,5,6,8,9,12,11)	[1,2,3,5,4], flag value is set to false, so break is executed and the program ends.	[1,2,3,5,4]	[1,2,3,5,4]	PASS
3	Enter array and verify bubble sort algorithm P3(1,2,3,11)	[1,2,3,4,5], this is the step when the whole array has been traversed and sorted, i.e.	[1,2,3,4,5]	[1,2,3,4,5]	PASS

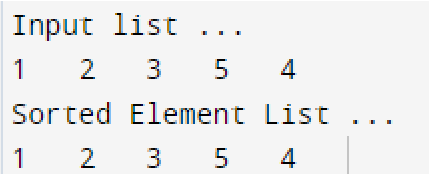
		var i has been decremented up to the length of the array			
4	Enter array and verify bubble sort algorithm P4(1,2,3,4,5,9,10,11)	[3,1,2,5,4], This is the case when 1st element is compared with all the other elements present in the array, so here 3, is compared with 1, then,2, then 5, if 3< 5 so next time 5 is checked with 4 and flag value is set to true	[1,2,3,4,5]	[1,2,3,4,5]	PASS
5	Enter array and verify bubble sort algorithm P5(1-2-3-4-5-9-12-11)	[1,2,3,4,5], when j=1 means one-time traversing has done and no flag is set	[1,2,3,4,5]	[1,2,3,4,5]	PASS

Test Case 1



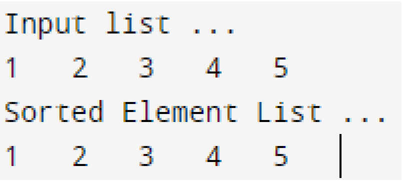
```
Input list ...
4    1    3    5    2
Sorted Element List ...
1    4    3    5    2
```

Test Case 2



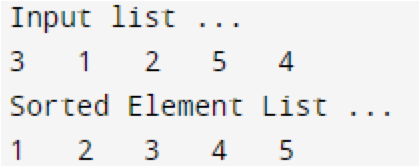
```
Input list ...
1    2    3    5    4
Sorted Element List ...
1    2    3    5    4
```

Test Case 3



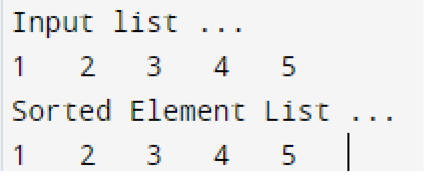
```
Input list ...
1    2    3    4    5
Sorted Element List ...
1    2    3    4    5
```

Test Case 4



```
Input list ...
3    1    2    5    4
Sorted Element List ...
1    2    3    4    5
```

Test Case 5



```
Input list ...
1    2    3    4    5
Sorted Element List ...
1    2    3    4    5
```