UNIVERSITY OF KARACHI

DEPARTMENT OF COMPUTER SCIENCE

(MORNING PROGRAMME)
MASTERS IN COMPUTER SCIENCE (FINAL) FIRST SEMESTER 2020
CS-623 DATA STRUCTURE AND ALGORITHM ANALYSIS

GRAND ASSIGNMENT 1

ARRAY

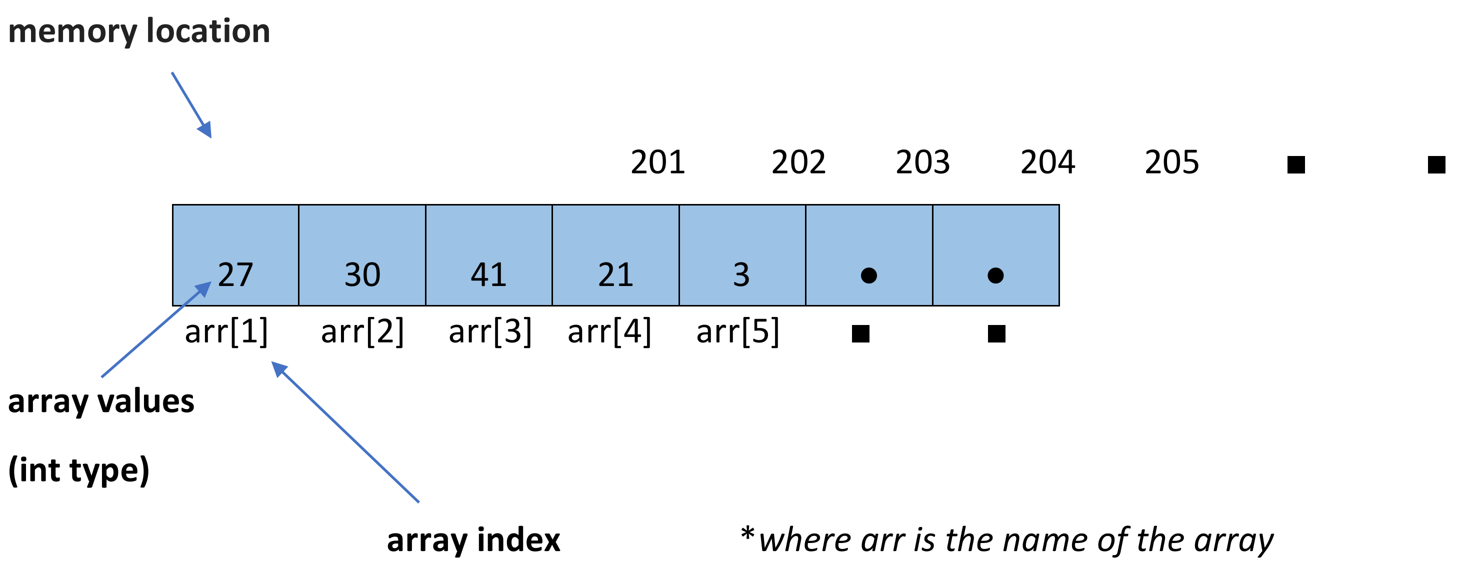
GROUP MEMBERS

STUDENT-ID	NAME OF STUDENT
P1810003	Afifa Aseer Humat
P1810004	Aiman Saira
P1810019	Hira Hassan Awan
P1810035	Rabab Zehra

INTRODUCTION TO ARRAYS

An array is collection of items stored at contiguous memory locations use to store multiple items of same type together.

A specific element in an **array** is accessed by an index.



Note:

Array's index is always start from '0' location but we choose to start its index from '1' location.

ADVANTAGES OF ARRAY:

1. Arrays represent multiple data items using same name.
2. In arrays, the elements can be accessed randomly by using the index numbers.
3. Arrays allocate memory in conrigious memory locations for all its elements. Hence there is no chamce of extra memory being allocated in case of arrays.this avoids memory overflow or shortage of memory.
4. Using arrays the other data structures like linked lists, stacks, queues, trees etc can be implimented.

APPLICATIONS OF ARRAY:

1. Array stores data elements of the same data type.

2. It maintains multiple varibale names or large data using/under a single name.This avoids the confusion of using multiple variables.
3. Arrays can be used for sorting data elements. Different Sorting techniques use arrays to store and sort elements easily.
4. Arrays can be used for CPU Scheduling.
5. They can be used for performing matrix operations.Many databases,small or large,one-dimensional or two-dimensional arrays whose elements are records.
6. Lastly, arrays are also used to implement other data structures like Stacks, Queues, Heaps, Hash tables etc.

ARRAY REAL WORLD EXAMPLE:

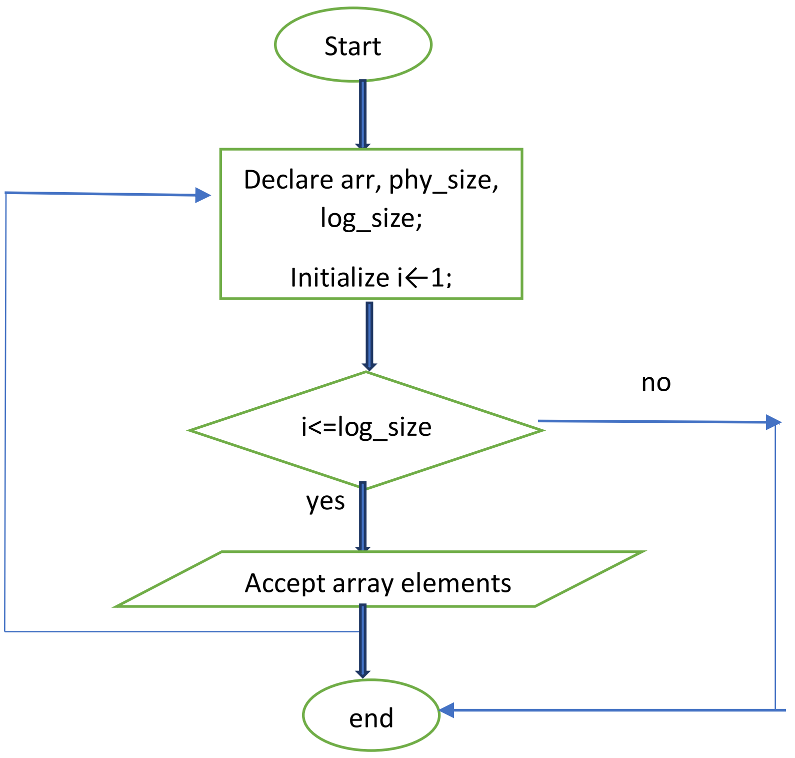
1. A quite trivial example would be your phone contacts. See the software will simply place your contacts in an array, use sorting to arrange them. Now each contact will have a unique index (acc. to its position), and there's it you can keep adding contacts, values will be updated and it will just be added to a single array. Do we need any variables? No just one array and we simply manipulate its positions.
2. Arrangement of ladder boards in games. Just a simple array that stores all the scores and sorts them in a descending manner.

ARRAY OPERATIONS:

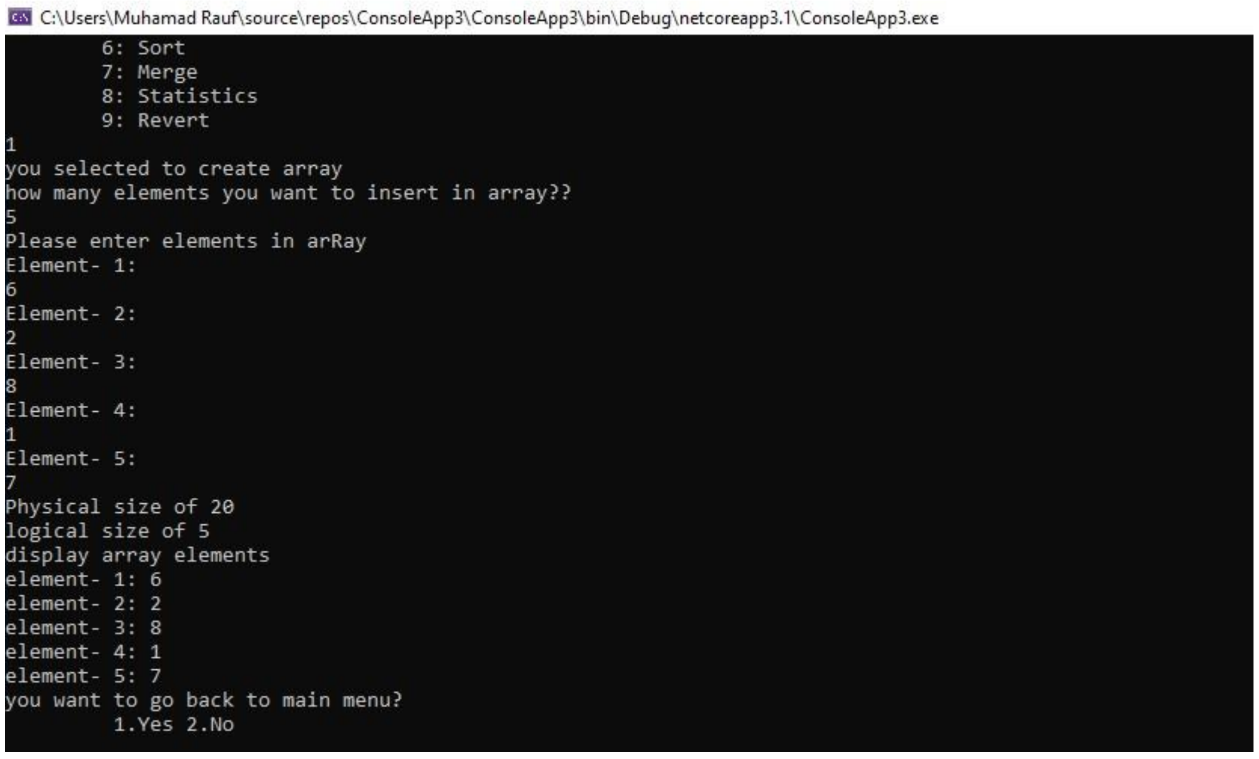
1. Create
2. Insert
3. Delete
4. Update
5. Search
6. Merge
7. Sort
8. Revert
9. Statistics

CREATE ARRAY:

FLOWCHART:



OUTPUT:

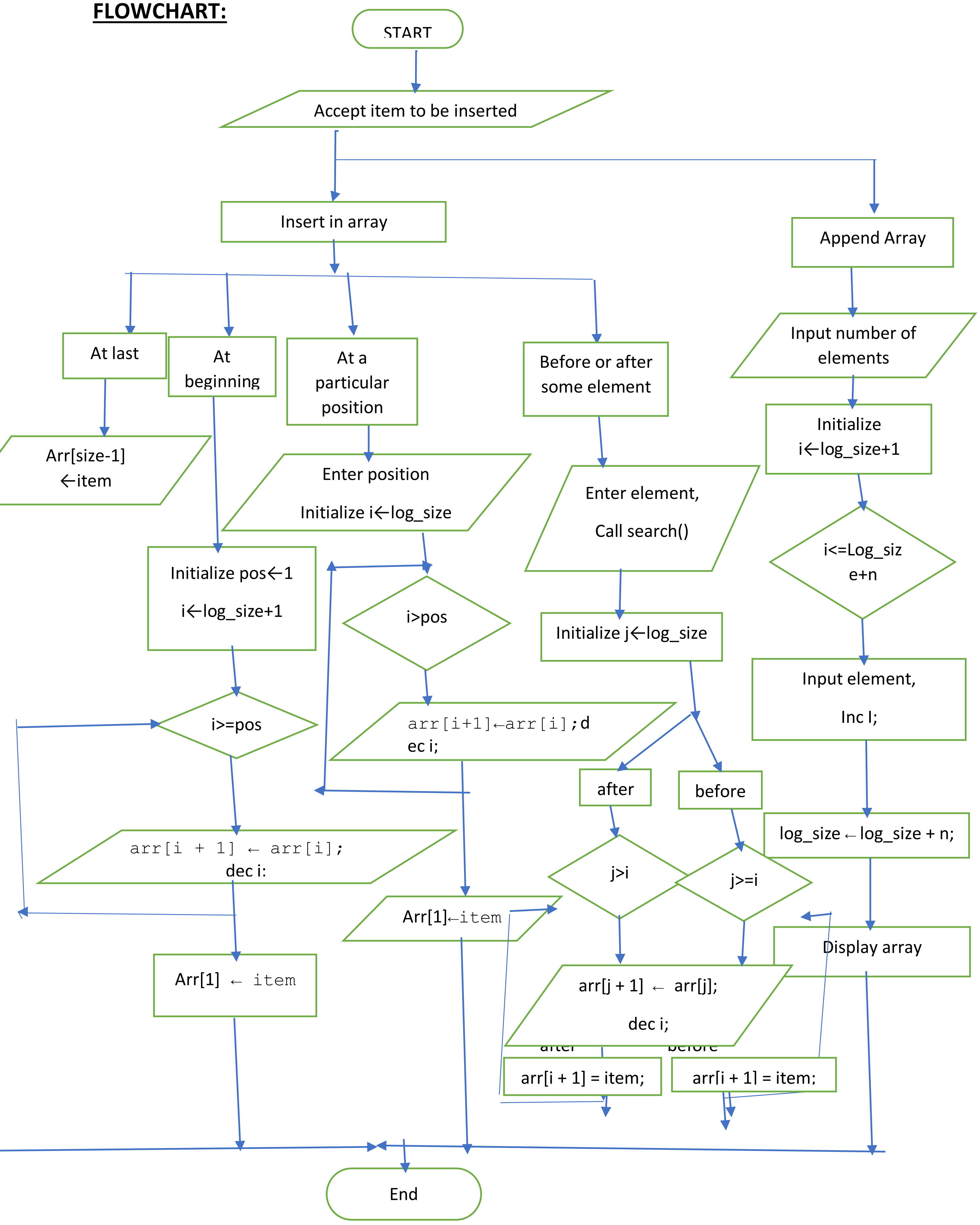


C:\Users\Muhamad Rauf\source\repos\ConsoleApp3\ConsoleApp3\bin\Debug\netcoreapp3.1\ConsoleApp3.exe

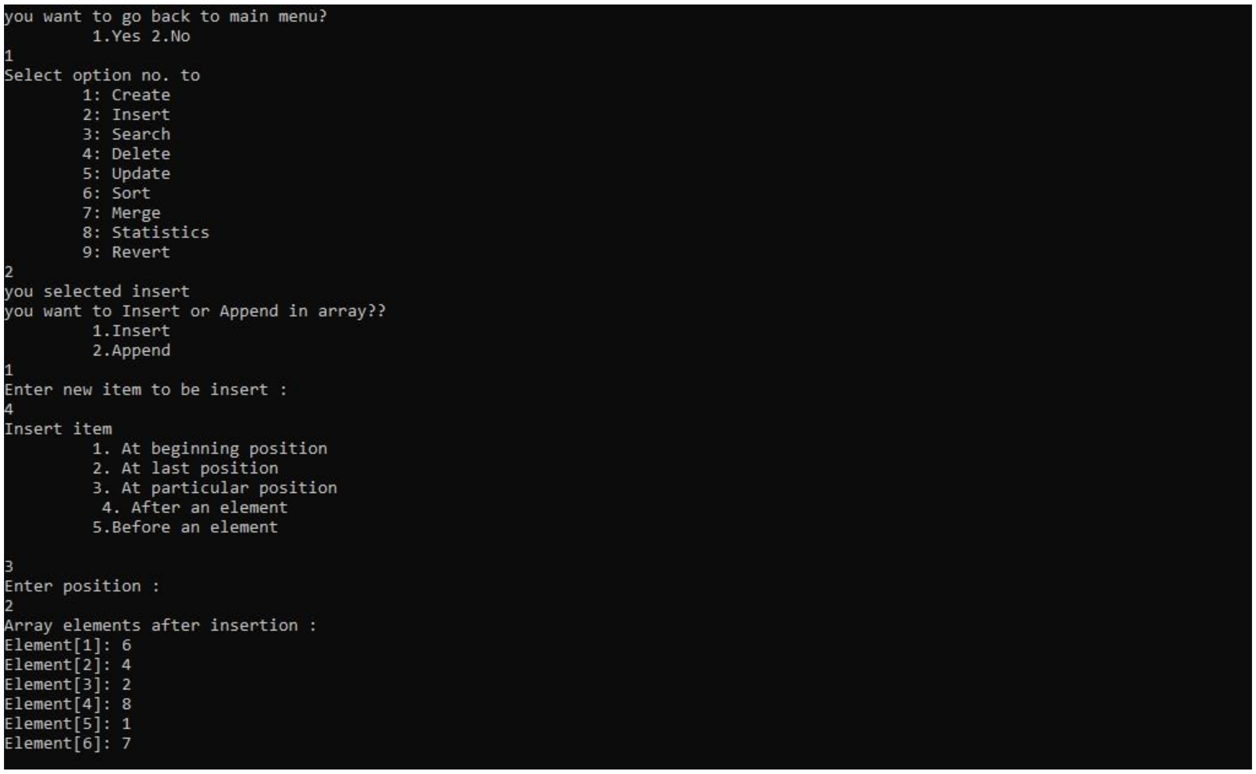
```
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
1
you selected to create array
how many elements you want to insert in array??
5
Please enter elements in arRay
Element- 1:
6
Element- 2:
2
Element- 3:
8
Element- 4:
1
Element- 5:
7
Physical size of 20
logical size of 5
display array elements
element- 1: 6
element- 2: 2
element- 3: 8
element- 4: 1
element- 5: 7
you want to go back to main menu?
         1.Yes 2.No
```

INSERT IN ARRAY:

FLOWCHART:



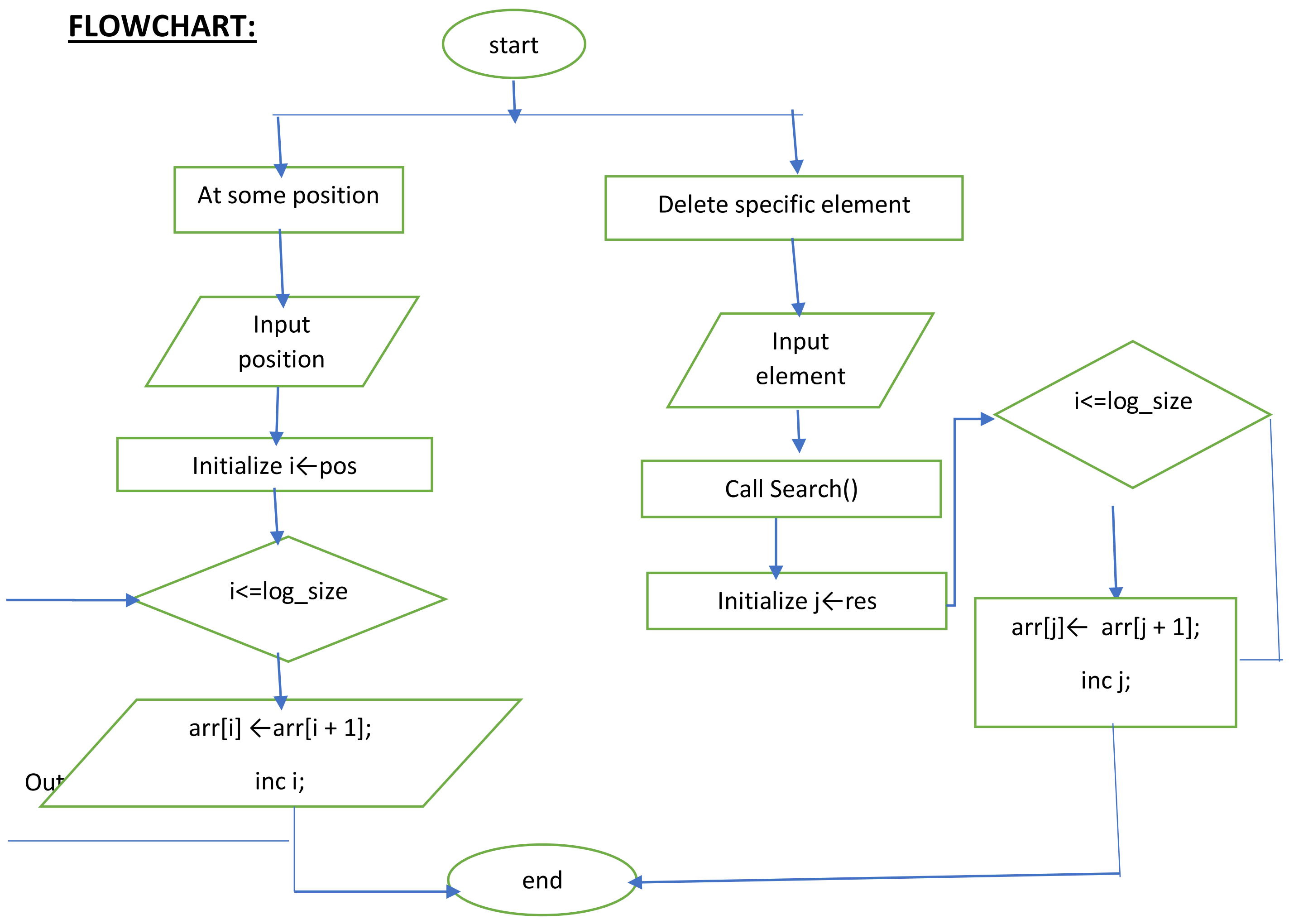
OUTPUT:



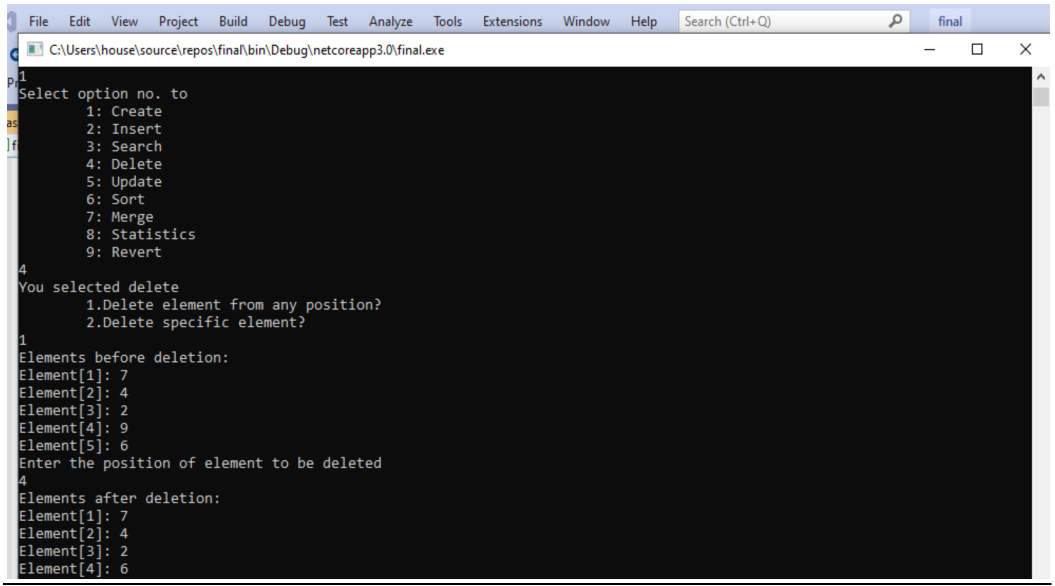
```
you want to go back to main menu?
         1.Yes 2.No
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
2
you selected insert
you want to Insert or Append in array??
         1.Insert
         2.Append
1
Enter new item to be insert :
4
Insert item
         1. At beginning position
         2. At last position
         3. At particular position
          4. After an element
         5.Before an element

3
Enter position :
2
Array elements after insertion :
Element[1]: 6
Element[2]: 4
Element[3]: 2
Element[4]: 8
Element[5]: 1
Element[6]: 7
```

DELETE IN ARRAY:

FLOWCHART:



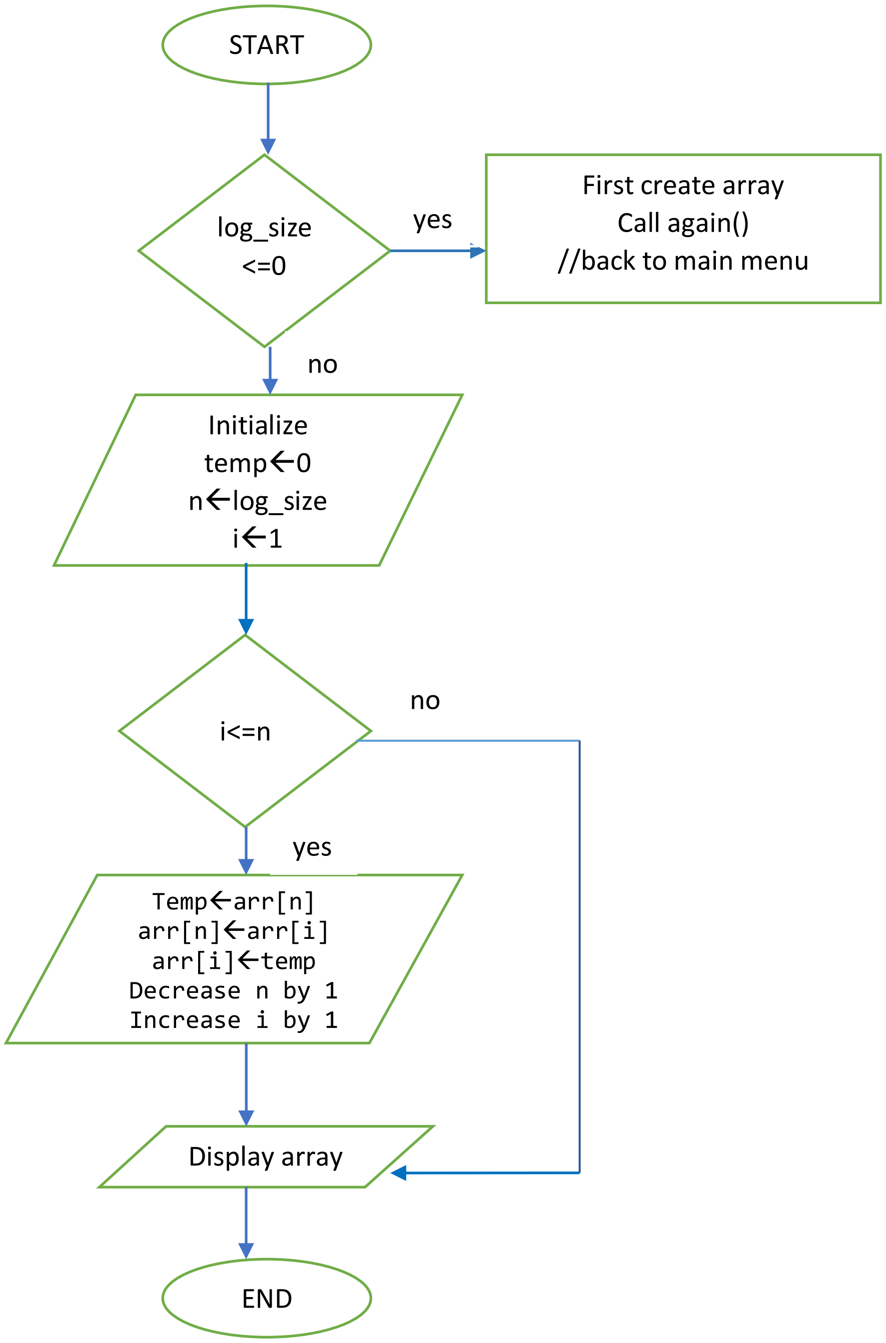
OUTPUT:


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
final
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe

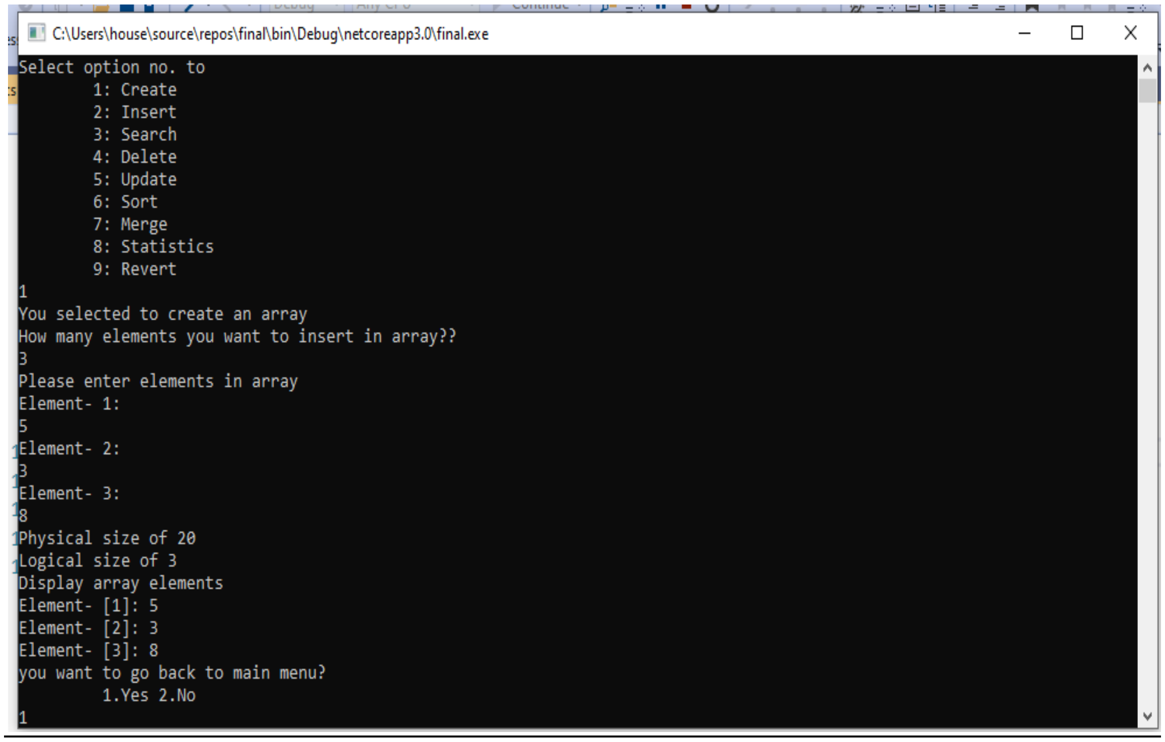
```
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
4
You selected delete
        1.Delete element from any position?
        2.Delete specific element?
1
Elements before deletion:
Element[1]: 7
Element[2]: 4
Element[3]: 2
Element[4]: 9
Element[5]: 6
Enter the position of element to be deleted
4
Elements after deletion:
Element[1]: 7
Element[2]: 4
Element[3]: 2
Element[4]: 6
```

REVERT ARRAY:

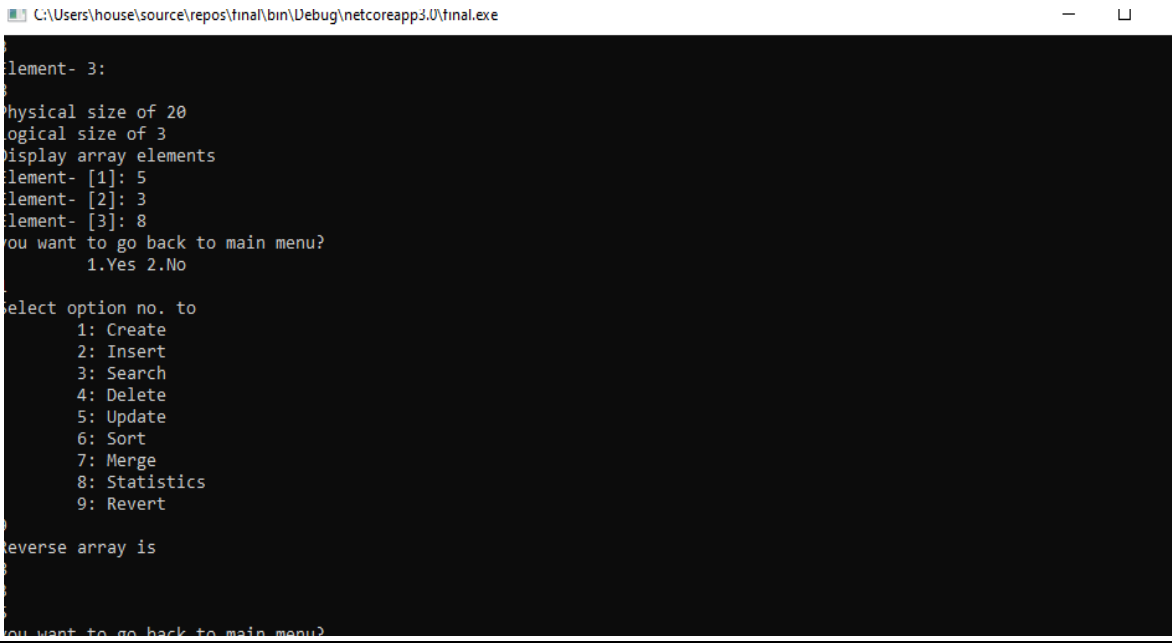
FLOWCHART:



OUTPUT:

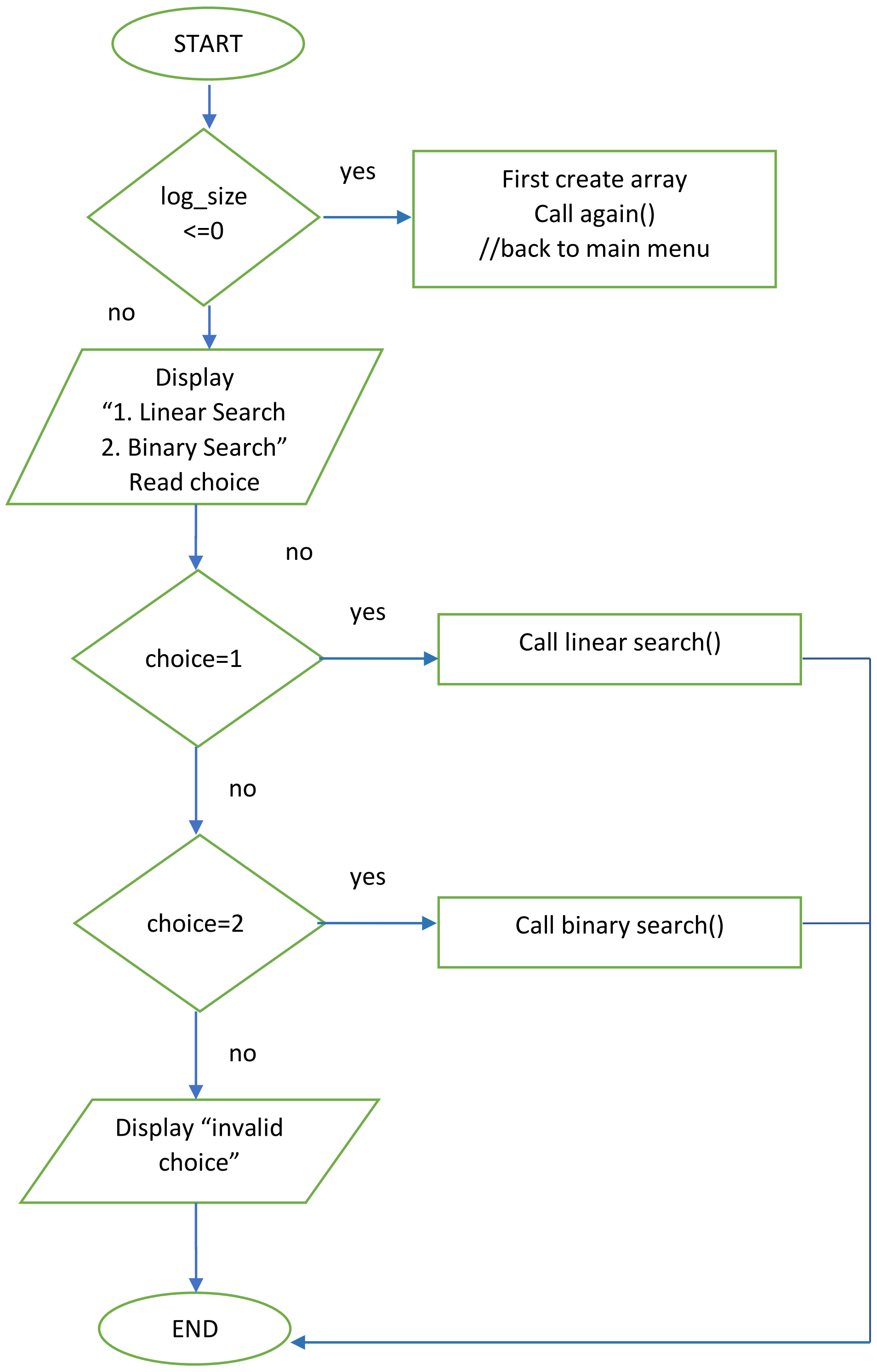


```
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
1
You selected to create an array
How many elements you want to insert in array??
3
Please enter elements in array
Element- 1:
5
Element- 2:
3
Element- 3:
8
Physical size of 20
Logical size of 3
Display array elements
Element- [1]: 5
Element- [2]: 3
Element- [3]: 8
you want to go back to main menu?
         1.Yes 2.No
1
```

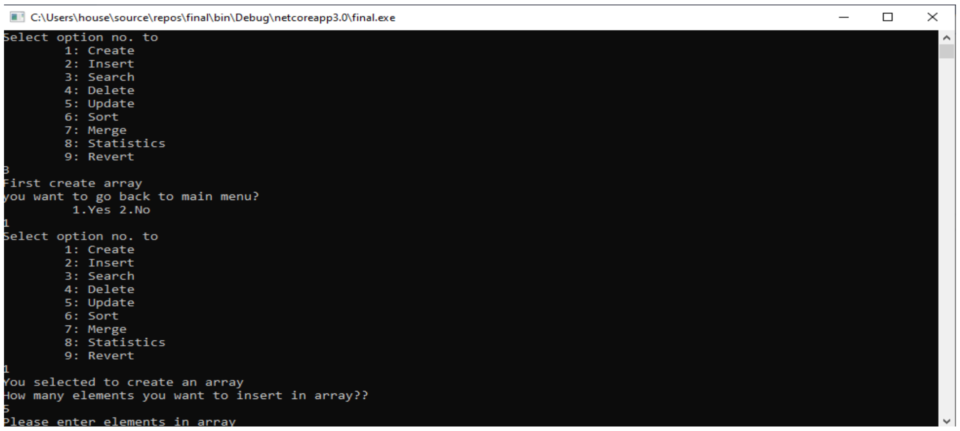


```
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe

lement- 3:

hysical size of 20
ogical size of 3
isplay array elements
lement- [1]: 5
lement- [2]: 3
lement- [3]: 8
ou want to go back to main menu?
        1.Yes 2.No

elect option no. to
       1: Create
       2: Insert
       3: Search
       4: Delete
       5: Update
       6: Sort
       7: Merge
       8: Statistics
       9: Revert

everse array is




ou want to go back to main menu?
```

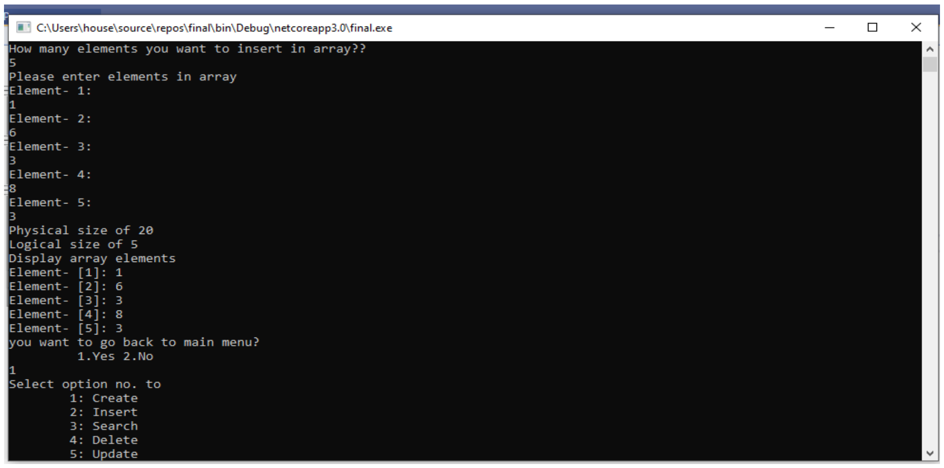
SEARCH IN ARRAY:



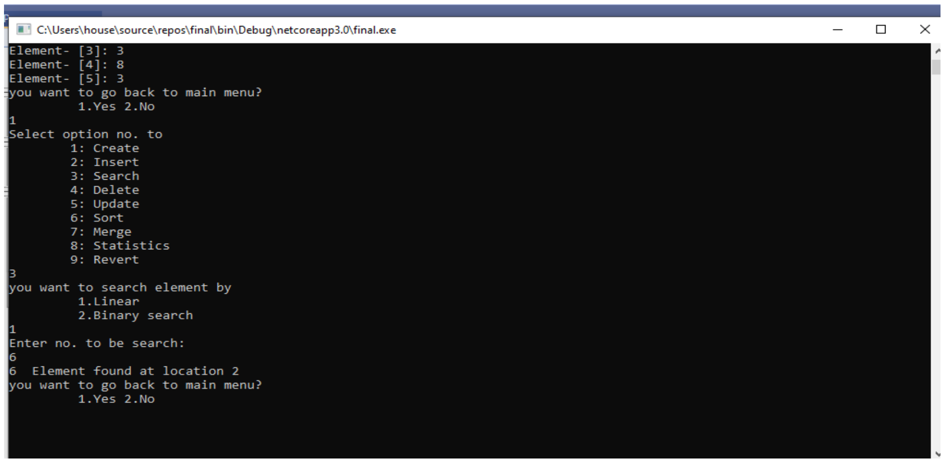
OUTPUT:



```
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
3
First create array
you want to go back to main menu?
         1.Yes 2.No
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
1
You selected to create an array
How many elements you want to insert in array??
5
Please enter elements in array
```



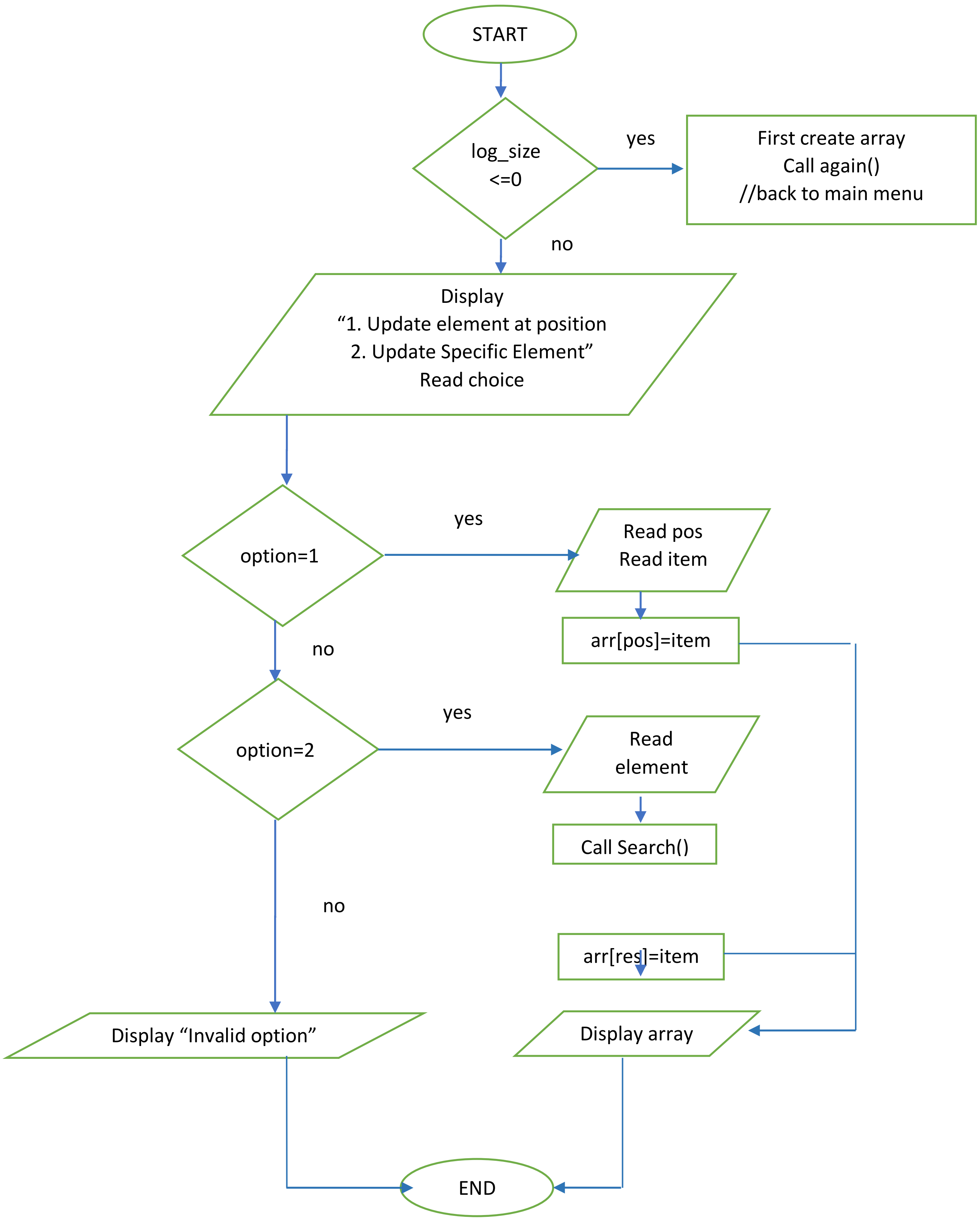
```
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe
How many elements you want to insert in array??
5
Please enter elements in array
Element- 1:
1
Element- 2:
6
Element- 3:
3
Element- 4:
8
Element- 5:
3
Physical size of 20
Logical size of 5
Display array elements
Element- [1]: 1
Element- [2]: 6
Element- [3]: 3
Element- [4]: 8
Element- [5]: 3
you want to go back to main menu?
         1.Yes 2.No
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
```



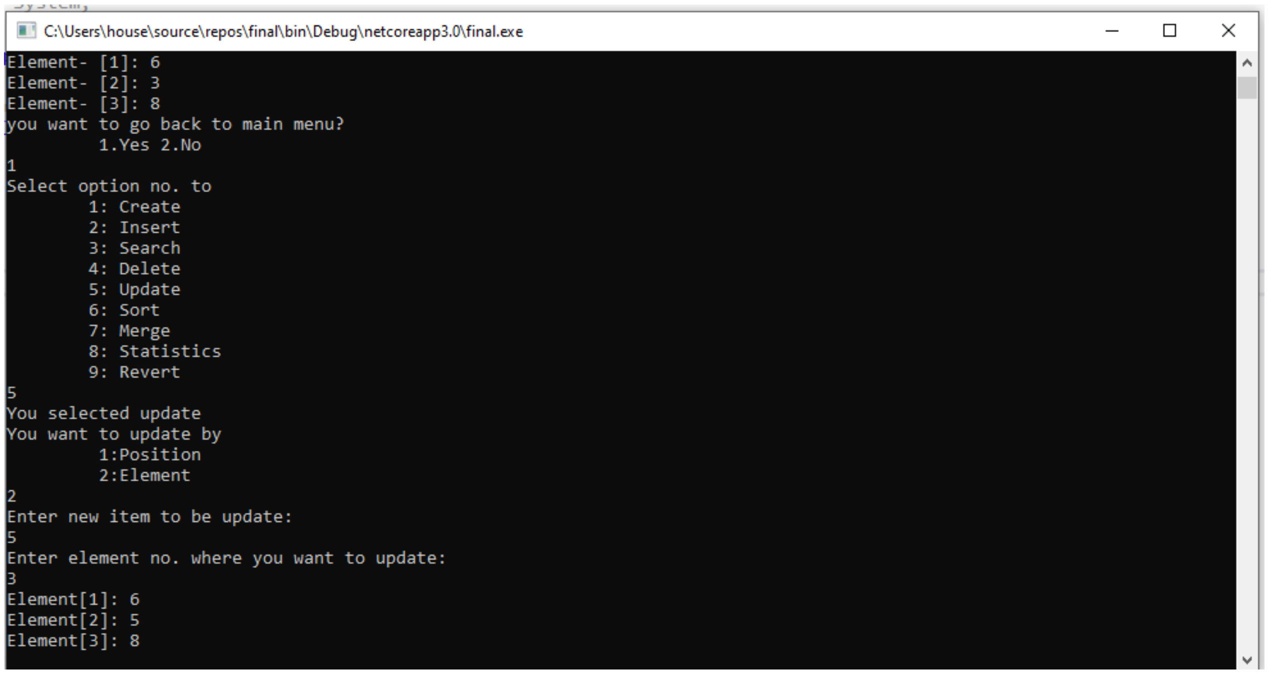
```
C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe
Element- [3]: 3
Element- [4]: 8
Element- [5]: 3
you want to go back to main menu?
         1.Yes 2.No
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
3
you want to search element by
         1.Linear
         2.Binary search
1
Enter no. to be search:
6
6  Element found at location 2
you want to go back to main menu?
         1.Yes 2.No
```

UPDATE IN ARRAY:

FLOWCHART:



OUTPUT:

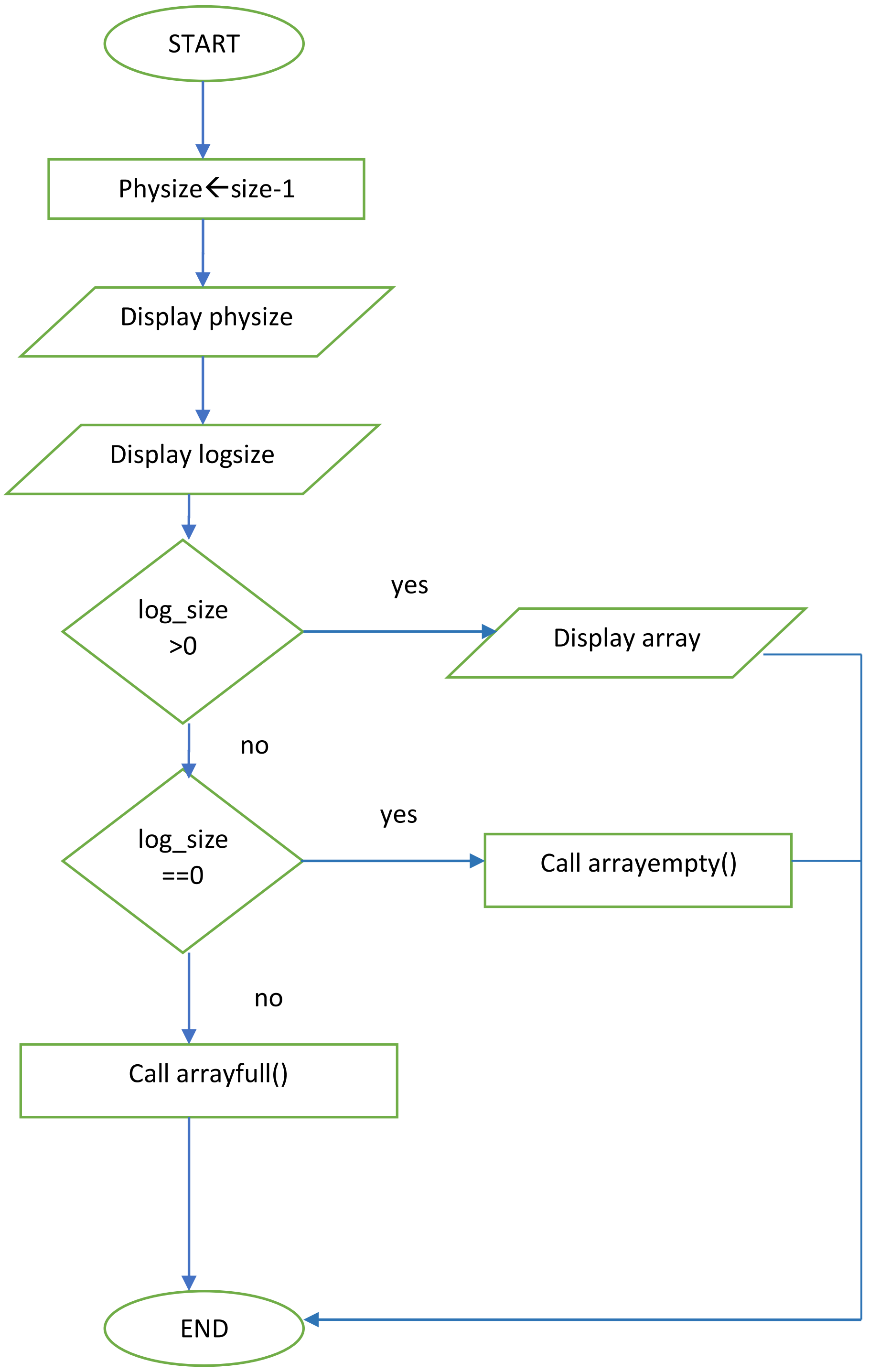


C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe

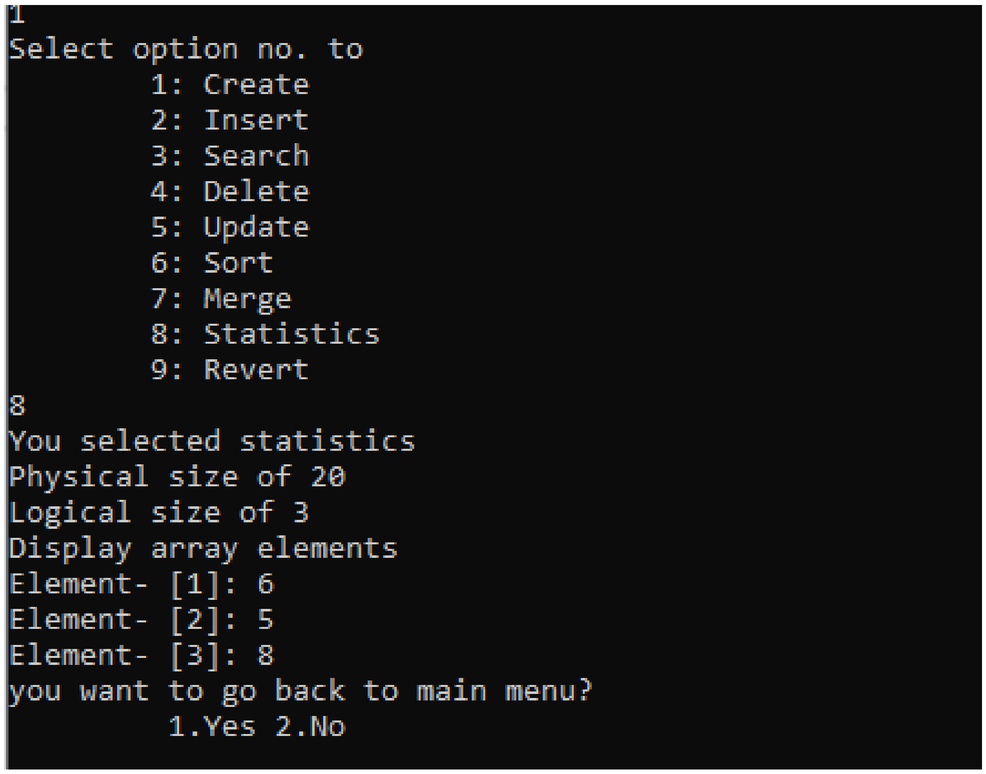
```
Element- [1]: 6
Element- [2]: 3
Element- [3]: 8
you want to go back to main menu?
         1.Yes 2.No
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
5
You selected update
You want to update by
         1:Position
         2:Element
2
Enter new item to be update:
5
Enter element no. where you want to update:
3
Element[1]: 6
Element[2]: 5
Element[3]: 8
```

STATISTICS OF ARRAY:

FLOWCHART:



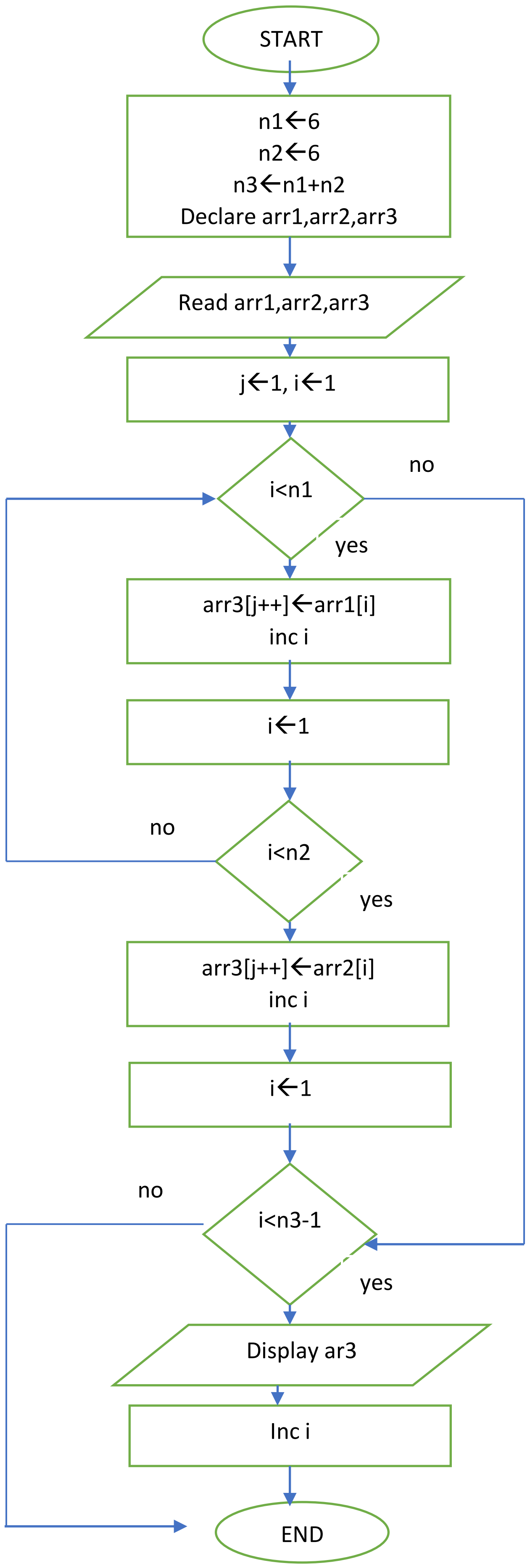
OUTPUT:



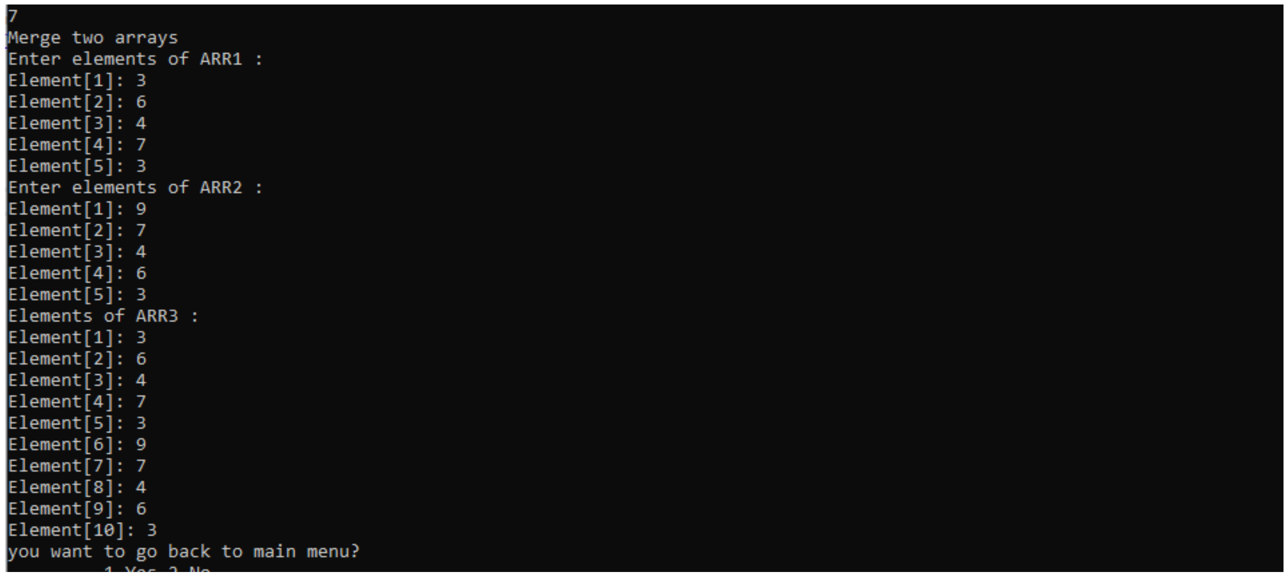
```
1
Select option no. to
        1: Create
        2: Insert
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
8
You selected statistics
Physical size of 20
Logical size of 3
Display array elements
Element- [1]: 6
Element- [2]: 5
Element- [3]: 8
you want to go back to main menu?
         1.Yes 2.No
```

MERGE ARRAYS:

FLOWCHART:



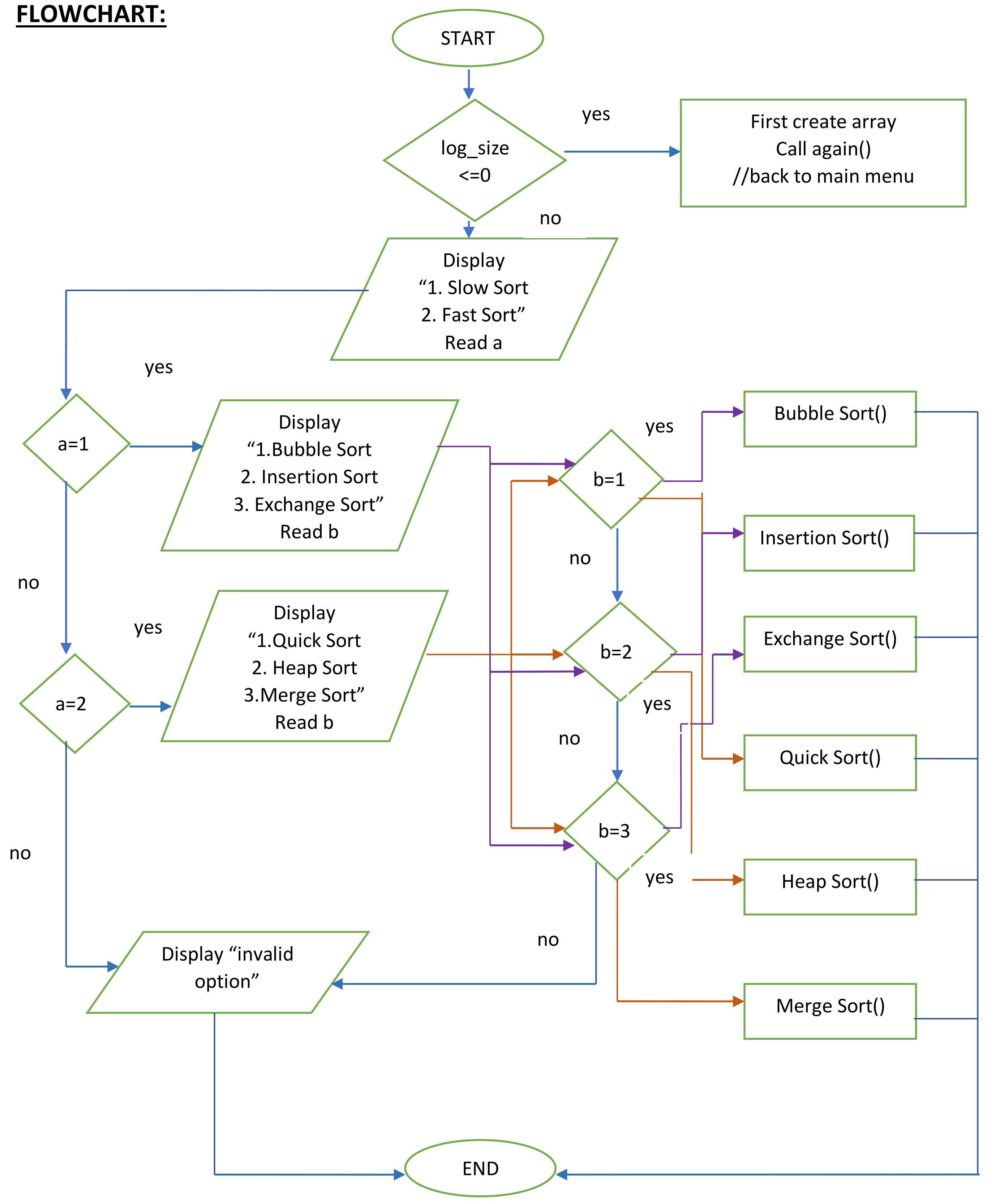
OUTPUT:



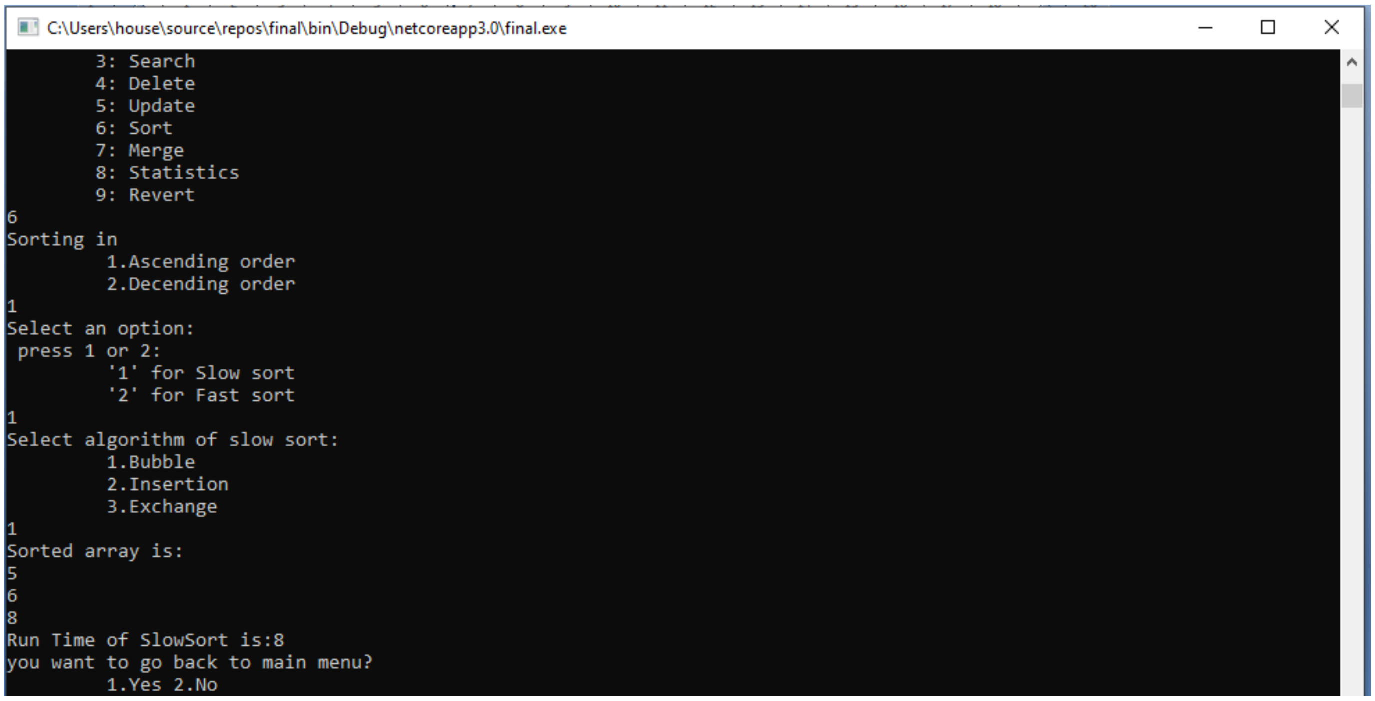
```
7
Merge two arrays
Enter elements of ARR1 :
Element[1]: 3
Element[2]: 6
Element[3]: 4
Element[4]: 7
Element[5]: 3
Enter elements of ARR2 :
Element[1]: 9
Element[2]: 7
Element[3]: 4
Element[4]: 6
Element[5]: 3
Elements of ARR3 :
Element[1]: 3
Element[2]: 6
Element[3]: 4
Element[4]: 7
Element[5]: 3
Element[6]: 9
Element[7]: 7
Element[8]: 4
Element[9]: 6
Element[10]: 3
you want to go back to main menu?
         1 Yes 2 No
```

SORT ARRAY:

FLOWCHART:



OUTPUT:



C:\Users\house\source\repos\final\bin\Debug\netcoreapp3.0\final.exe

```
        3: Search
        4: Delete
        5: Update
        6: Sort
        7: Merge
        8: Statistics
        9: Revert
6
Sorting in
         1.Ascending order
         2.Decending order
1
Select an option:
 press 1 or 2:
         '1' for Slow sort
         '2' for Fast sort
1
Select algorithm of slow sort:
         1.Bubble
         2.Insertion
         3.Exchange
1
Sorted array is:
5
6
8
Run Time of SlowSort is:8
you want to go back to main menu?
         1.Yes 2.No
```

FUNCTIONAL FLOW DIAGRAM:

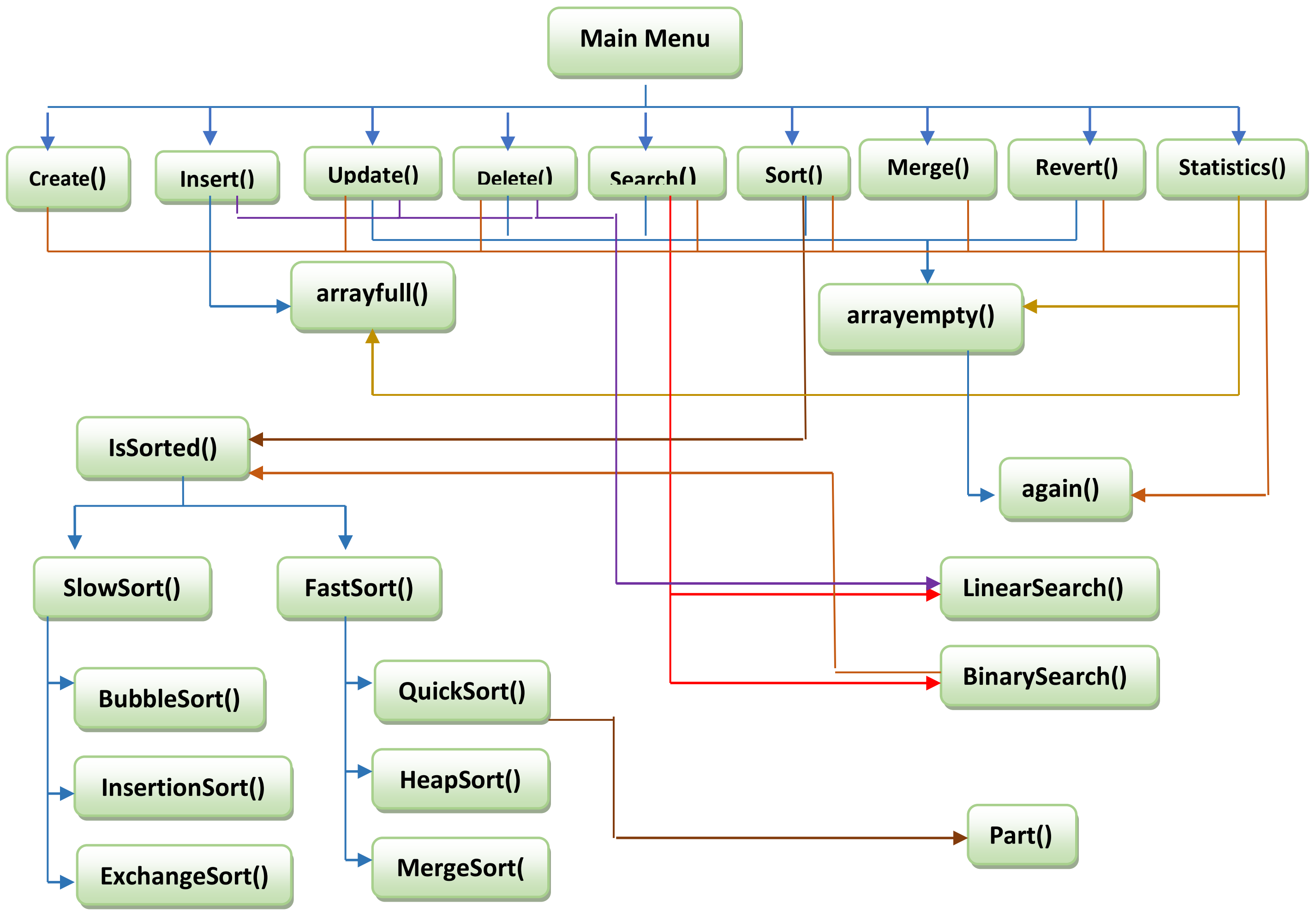


TABLE:

Algorithm	Fast/Slow?	Run time Complexity		Space time Complexity		In place?	Stable
		BC	WC	BC	WC	Yes/No	Yes/No?
Linear Search	Slow	O(n)	O(n)	–	O(1)	–	–
Binary Search	Fast	O(n)	O(log(n))	–	O(1)	–	–
Bubble Sort	Slow	O(n)	$O(n^2)$	–	O(1)	Yes	Yes
Insertion Sort	Slow	O(n)	$O(n^2)$	–	O(1)	Yes	Yes
Selection Sort	Slow	$O(n^2)$	$O(n^2)$	–	O(1)	Yes	No
Quick Sort	Fast	O(nlog(n))	$O(n^2)$	–	O(log(n))	Yes	No
Heap Sort	Fast	O(nlog(n))	O(nlog(n))	–	O(1)	Yes	No
Merge Sort	Fast	O(nlog(n))	O(nlog(n))	–	O(n)	No	Yes

ANNEXURE:

```
//main class

using System;

namespace ConsoleApp8
{
    class Program
    {
        static void Main(string[] args)
        {
            Array obj = new Array();
            obj.Menu();
        }
    }
}
```

```
// second class

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp8

{

  class Array
```

```
{
   int ts = 0, tf = 0;
   int size = 21;
   int log_size, i, search;
   int[] arr = new int[21];
   int res = 0;
   int phy_size;
   public void Menu()    //Method of menu
   {
      int n;
      Console.WriteLine("Select option no. to \n\t1: Create \n\t2: Insert \n\t3: Search \n\t" +
         "4: Delete \n\t5: Update \n\t6: Sort\n\t7: Merge \n\t8: Statistics \n\t9: Revert");
      n = Convert.ToInt16(Console.ReadLine());
      switch (n)
      {
         case 1:
            Console.WriteLine("You selected to create an array");
            Create();
            again();
            break;
         case 2:
            Console.WriteLine("You selected insert");
            Insert();
            again();
            break;
         case 3:
            search1();
            again();
            break;
         case 4:
            Console.WriteLine("You selected delete");
            Delete();
```

```
    again();
    break;
case 5:
    Console.WriteLine("You selected update");
    Update();
    again();
    break;
case 6:
    Console.WriteLine("Sorting in \n\t 1.Ascending order \n\t 2.Decending order");
    int sor = Convert.ToInt16(Console.ReadLine());
    if (sor == 1)
    {
        Sort();
    }
    if (sor == 2)
    {
        Sort();
        Revert();
    }
    again();
    break;
case 7:
    Console.WriteLine("Merge two arrays");
    MergeArr();
    again();
    break;
case 8:
    Console.WriteLine("You selected statistics");
    Statistics();
    again();
    break;
case 9:
```

```
            Console.WriteLine("Reverse array is");
            Revert();
            again();
            break;



    }


}



public int Create()    //Method to create array
{
    phy_size = size - 1;
    Console.WriteLine("How many elements you want to insert in array??");
    log_size = Convert.ToInt16(Console.ReadLine());
    if (log_size >= size)
        Console.WriteLine("Out of range");
    else
    {
        Console.WriteLine("Please enter elements in array");
        for (int i = 1; i <= log_size; i++)
        {
            Console.WriteLine("Element- {0}: ", i);
            arr[i] = Convert.ToInt16(Console.ReadLine());
        }
    }
    Statistics();
    return 1;
}
public void Insert() //Method to insert elements in array
{
```

```
if (log_size <= 0)
{
   Console.WriteLine("First create array");
   again();
}
else if (log_size > 0)
{
   int choice;
   Console.WriteLine("You want to Insert or Append in array?? \n\t 1.Insert \n\t 2.Append ");
   choice = Convert.ToInt16(Console.ReadLine());
   if (choice == 1)
   {
      int pos, item;
      log_size = log_size + 1;
      Console.WriteLine("Insert item \n\t 1. At beginning position\n\t 2. At last position\n\t " +
                   "3. At particular position\n\t 4. After an element\n\t 5. Before an element");
      int m = int.Parse(Console.ReadLine());
      Console.WriteLine("Enter new item to be insert : ");
      item = int.Parse(Console.ReadLine());
      if (m == 1)
      {
         for (i = log_size; i >= 1; i--)
         {
            arr[i + 1] = arr[i];
         }
         arr[1] = item;

      }
      else if (m == 2)
      {
         arr[log_size] = item;
      }
```

```
else if (m == 3)
{
    Console.WriteLine("Enter position : ");
    pos = int.Parse(Console.ReadLine());
    //Perform shift opearation
    for (i = log_size; i >= pos; i--)
    {
        arr[i + 1] = arr[i];
    }
    arr[pos] = item;
}
else if (m == 4)
{
    Console.WriteLine("Enter the element where after you want to insert new element");
    linear();
    for (i = log_size; i >= res; i--)
    {
        arr[i + 1] = arr[i];
    }
    arr[res + 1] = item;
}
else if (m == 5)
{
    Console.WriteLine("Enter the element where before you want to insert new element:");
    linear();
    for (i = log_size; i > res - 1; i--)
    {
        arr[i + 1] = arr[i];
    }
    arr[res] = item;
}
//print array after insertion
```

```
            Console.WriteLine("Array elements after insertion : ");
            for (i = 1; i <= log_size; i++)
            {
                Console.WriteLine("Element[" + (i) + "]: " + arr[i]);
            }
        }
        else if (choice == 2)
        {
            Console.WriteLine("How many elements you wants to insert??");
            int n = Convert.ToInt16(Console.ReadLine());
            for (i = log_size + 1; i <= (log_size + n); i++)
            {
                Console.WriteLine("Element- {0}: ", i);
                arr[i] = Convert.ToInt16(Console.ReadLine());
            }
            //print array after insertion
            Console.WriteLine("Array elements after insertion : ");
            log_size = log_size + n;
            for (i = 1; i <= log_size; i++)
            {
                Console.WriteLine("Element[" + (i) + "]: " + arr[i]);
            }
        }
        Console.ReadLine();
    }
}
public int search1() //Method to search element
{
    if (log_size <= 0)
    {
        Console.WriteLine("First create array");
        again();
    }
    else if (log_size > 0)
    {
        int choice;
        Console.WriteLine("you want to search element by \n\t 1.Linear \n\t 2.Binary search ");
        choice = Convert.ToInt16(Console.ReadLine());
        Console.WriteLine("Enter no. to be search: ");
        if (choice == 1)
        {
            linear();
            Console.WriteLine("{0}  Element found at location {1}", search, res);
        }
        else if (choice == 2)
        {
            search = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("For Binary Search array must be sorted");
            isSorted();
        }
    }
    return 1;
}
public void linear() //Method of linear search
{
    search = Convert.ToInt32(Console.ReadLine());
    for (i = 1; i <= log_size; i++)
    {
        if (arr[i] == search)
        {
            res = i;
        }
    }
    if (res == 0)
    {
        Console.WriteLine("Element not found");
    }
}
public int isSorted()  //Method to check array is sorted or not
{
    i = 0;
    int c = 1, d = 1;

    while ((c == 1 || d == 1) && i < log_size - 1)
    {
        if (arr[i] < arr[i + 1])
        {
            c = 0;
        }

        else if (arr[i] > arr[i + 1])
        {
            d = 0;
        }
        i++;
    }
    if (d == 1)
    {
        Console.WriteLine("\tArray is sorted in Ascending order");
        binarysearch();
    }
    else if (c == 1)
    {
        Console.WriteLine("\tArray is sorted in Descending order");
        binarysearch();
    }
```

```
   else
   {
      Console.WriteLine("Array is not sorted \n\tSORRY! Binary search can not be apply:(");
   }
   return 0;
}


public int binarysearch()   //Method of binary search
{
   int beg, end, mid;
   beg = 1;
   end = log_size;
   while (beg <= end)
   {
      mid = (beg + end) / 2;
      if (search < arr[mid])
         end = mid - 1;
      else if (search > arr[mid])
         beg = mid + 1;
      else if (search == arr[mid])
      {
         Console.WriteLine("Item Found");
         Console.WriteLine("Element {0} found at location {1}\n", search, mid);
         break;
      }
   }
   return 1;
}

public int Delete()  //Method to delete element
{
```

```
if (log_size <= 0)
{
    Console.WriteLine("First create array");
    again();
}
else if (log_size > 0)
{
    int pos, select;
    Console.WriteLine("\t1.Delete element from any position?\n\t2.Delete specific element?");
    select = Convert.ToInt32(Console.ReadLine());

    log_size = log_size - 1;
    switch (select)
    {
        case 1:
            Console.WriteLine("Enter the position of element to be deleted");
            pos = Convert.ToInt32(Console.ReadLine());
            for (i = pos; i <= log_size; i++)
            {
                arr[i] = arr[i + 1];
            }
            break;
        case 2:
            Console.WriteLine("Enter the element to be deleted");
            linear();
            for (int j = res; j <= log_size; j++)
            {
                arr[j] = arr[j + 1];
            }
            break;
    }
    Console.WriteLine("Elements after deletion: ");
```

```
            for (i = 1; i <= log_size; i++)
            {
                Console.WriteLine("Element[" + (i) + "]: " + arr[i]);
            }
            Console.ReadKey();
        }
        return 1;
    }
    public void Update()    //Method to update element in array
    {
        if (log_size <= 0)
        {
            Console.WriteLine("First create array");
            again();
        }
        else if (log_size > 0)
        {
            int pos, item;
            Console.WriteLine("You want to update by \n\t 1:Position \n\t 2:Element");
            int choice = Convert.ToInt16(Console.ReadLine());
            Console.WriteLine("Enter new item to be update: ");
            item = int.Parse(Console.ReadLine());
            if (choice == 1)
            {
                Console.Write("Enter postion no. to be update: ");
                pos = int.Parse(Console.ReadLine());
                arr[pos] = item;
            }
            else if (choice == 2)
            {
                Console.WriteLine("Enter element no. where you want to update: ");
                linear();
                arr[res] = item;
            }
            else
            {
                Console.WriteLine("you choose invalid option");
            }
            for (i = 1; i <= log_size; i++)
            {
                Console.WriteLine("Element[" + (i) + "]: " + arr[i]);
            }
            Console.ReadLine();
        }
    }
    public void Sort()  //Method to sort array
    {
        if (log_size <= 0)
        {
            Console.WriteLine("First create array");
            again();
        }
        else
        {
            int a, b;



            Console.WriteLine("Select an option:\n press 1 or 2: \n\t '1' for Slow sort \n\t '2' for Fast sort");
            a = Convert.ToInt16(Console.ReadLine());
            switch (a)
            {
                case 1:
                    Console.WriteLine("Select algorithm of slow sort:\n\t 1.Bubble\n\t 2.Insertion\n\t 3.Exchange");
                    b = Convert.ToInt16(Console.ReadLine());
                    if (b == 1 || b == 2 || b == 3)
                    {
                        BubbleSort();
                        Console.WriteLine("Run Time of SlowSort is:" + ts);
                    }
                    else
                    {
                        Console.WriteLine("You selected a non existing option");
                    }
                    break;
                case 2:
                    Console.WriteLine("Select type of fast sort:\n 1.Quick\n 2.Heap\n 3.Merge");
                    b = Convert.ToInt16(Console.ReadLine());
                    if (b == 1 || b == 2 || b == 3)
                    {
                        int left = 1;
                        QuickSort(arr, left, log_size);
                        //Print Sorted array
                        Console.WriteLine("Sorted array is:");
                        for (int i = 1; i <= log_size; i++)
                        {
                            Console.WriteLine(+arr[i]);
                        }
                        Console.WriteLine("RunTime of FastSort is:" + tf);
                    }
                    else
                    {
                        Console.WriteLine("you selected a non existing option");
                    }
                    break;

            }
```

```
    }
}
public int BubbleSort()  //Method of bubble sort
{
    int swap;
    ts = ts + 1;
    for (int i = 1; i < log_size; i++)
    {
        for (int j = 1; j <= log_size - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                swap = arr[j + 1];
                ts = ts + 2;
                arr[j + 1] = arr[j];
                ts = ts + 2;
                arr[j] = swap;
                ts = ts + 2;
            }
        }
    }
    //Print Sorted array
    Console.WriteLine("Sorted array is:");
    for (int i = 1; i <= log_size; i++)
    {
        Console.WriteLine(+arr[i]);
    }
    ts = ts + 1;
    return 1;
}
public int Part(int[] arr, int left, int right)
{
    int pivot;
    tf = tf + 1;
    pivot = arr[left];
    tf = tf + 2;
    while (true)
    {
        while (arr[left] < pivot)
        {
            left++;
            tf = tf + 1;
        }
        while (arr[right] > pivot)
        {
            right--;
            tf = tf + 1;
        }
        if (left < right)
        {
            int temp = arr[right];
            tf = tf + 2;
            arr[right] = arr[left];
            tf = tf + 2;
            arr[left] = temp;
            tf = tf + 2;
        }
        else
        {
            return right;
        }
        tf = tf + 1;
    }
}
```

```
public void QuickSort(int[] arr, int left, int right)
{
   int pivot; tf = tf + 1;
   if (left < right)
   {
      pivot = Part(arr, left, right);
      if (pivot > 1)
      {
         QuickSort(arr, left, pivot - 1);
      }
      if (pivot + 1 < right)
      {
         QuickSort(arr, pivot + 1, right);
      }
   }

}
public void MergeArr()   //Method of merge sort
{
   int n1 = 6, n2 = 6;
   int n3 = n1 + n2;
   int j = 1, i = 1;
   int[] arrA = new int[n1];
   int[] arrB = new int[n2];
   int[] arrC = new int[n3];
   Console.WriteLine("Enter elements of ARR1 : ");
   for (i = 1; i < n1; i++)
   {
      Console.Write("Element[" + (i) + "]: ");
      arrA[i] = Convert.ToInt32(Console.ReadLine());
   }
   Console.WriteLine("Enter elements of ARR2 : ");
```

```
    for (i = 1; i < n2; i++)
    {
        Console.Write("Element[" + (i) + "]: ");
        arrB[i] = int.Parse(Console.ReadLine());
    }
    //Merge arr1 and arr2 to arr3
    for (i = 1, j = 1; i < n1; i++)
    {
        arrC[j++] = arrA[i];
    }
    for (i = 1; i < n2; i++)
    {
        arrC[j++] = arrB[i];
    }
    //Print merged array
    Console.WriteLine("Elements of ARR3 : ");
    for (i = 1; i < n3 - 1; i++)
    {
        Console.WriteLine("Element[" + (i) + "]: " + arrC[i]);
    }
}
public void Revert()  //Method to reverse array
{
    if (log_size <= 0)
    {
        Console.WriteLine("First create array");
        again();

    }
    else if (log_size > 0)
    {
        int temp = 0;
```

```
            int n = log_size;
            for (i = 1; i <= n; i++)
            {
                temp = arr[n];
                arr[n] = arr[i];
                arr[i] = temp;
                n--;
            }
            for (i = 1; i <= log_size; i++)
            {
                Console.WriteLine(arr[i]);
            }
        }
    }
    public int Statistics() // Method of array statistics
    {
        int physize = (size - 1);
        Console.WriteLine("Physical size of " + physize);
        Console.WriteLine("Logical size of " + log_size);
        if (log_size > 0)
        {
            Console.WriteLine("Display array elements");
            for (int i = 1; i <= log_size; i++)
            {
                Console.WriteLine("Element- [{0}]: {1} ", i, arr[i]);
            }
        }
        else if (log_size == 0)
        {
            arrayempty();
        }
        else
        {
            arrayfull();
        }
        return 1;
    }
    public int arrayfull()   //Method of array full
    {
        if (log_size == phy_size)
        {
            Console.WriteLine("array is full");
        }

        return 1;

    }

    public int arrayempty()  //Method of array empty
    {
        Console.WriteLine("you can insert element/s, array is not full");
        return 1;
    }
    public int again()  //Method of again go to menu
    {
        int j;
        Console.WriteLine("you want to go back to main menu?\n\t 1.Yes 2.No ");
        j = Convert.ToInt32(Console.ReadLine());
        if (j == 1)
            Menu();
        else if (j == 2)
            return 0;
        else Console.WriteLine("Incorrect command");
        return 1; }}}
```

THE END

X------X-----X------X