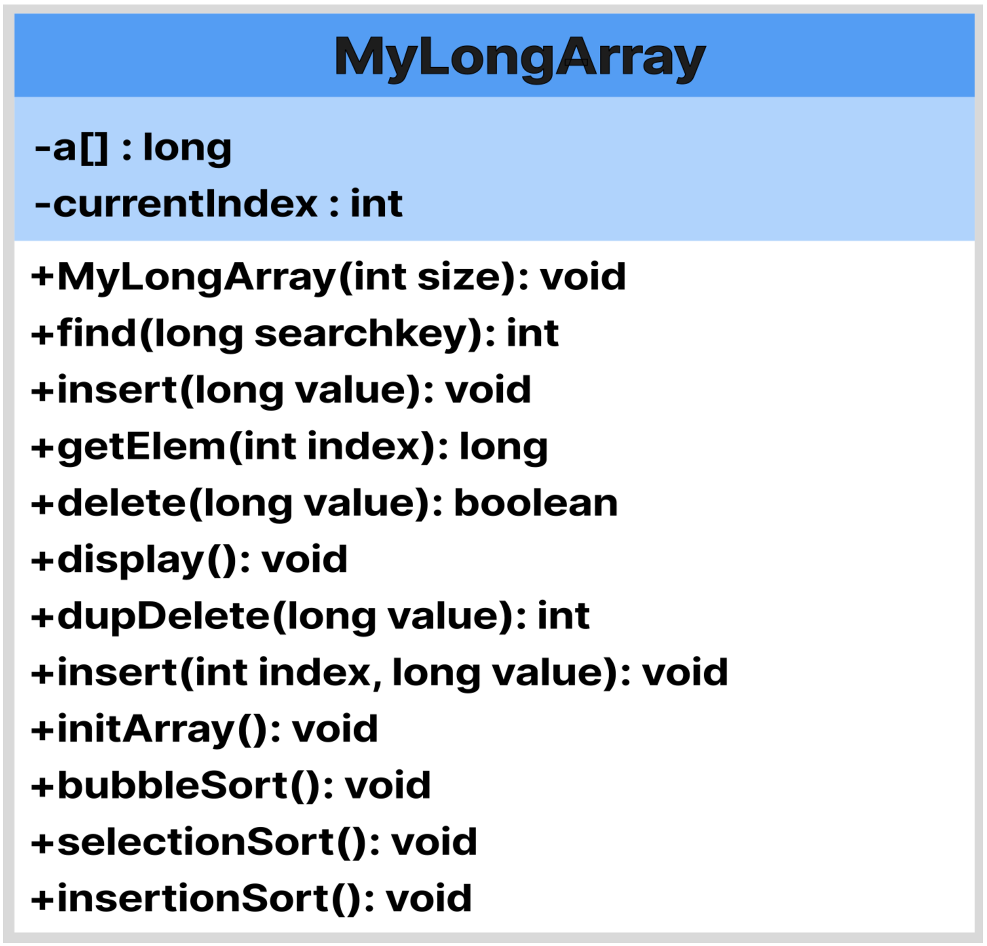
Laboratory-3

Experiment: Extend your long integer array class with the following new operations=
initArray(): void – Initialize all elements with random value [java.util.Random],
bubbleSort(),
selectionSort(),
insertionSort().
Write a test application for your class to demonstrate above operations.

Description of different Java constructs you used for the implementation:
The `initArray()` method of `MyLongArray` class uses the `nextLong()` method of the class `java.util.Random` to initialize the array with random long values.

Class Diagram:


MyLongArray
-a[] : long
-currentIndex : int
+MyLongArray(int size): void
+find(long searchkey): int
+insert(long value): void
+getElem(int index): long
+delete(long value): boolean
+display(): void
+dupDelete(long value): int
+insert(int index, long value): void
+initArray(): void
+bubbleSort(): void
+selectionSort(): void
+insertionSort(): void

Code:

```
import java.util.Scanner;
import java.util.Random;
/**
* This class is MyLongArray with attributes a, currentIndex and methods find(),
insert(), getElem(),
delete(), dupDelete(), display(), insert(), deleteAt(), initArray(), bubbleSort(),
selectionSort(), insertionSort().
* @author  Abhi Mehta
* @version 1.0.0
* @since 24-09-2023
*/
class MyLongArray {
long[] a;
int currentIndex;
/**
* Constructor of MyLongArray which initializes array and current index
* @param size a user defined input to initialize array capacity
*/
public MyLongArray(int size) {
a = new long[size];
currentIndex = 0;
}
/**
* This method calculates the remaining spaces available in the array.
* @return Integer value which indicates the space remaining.
*/
public int getRemainingPositions() {
return (a.length - (currentIndex + 1)) + 1;
}
/**
* This method finds the user given value in the array and returns it.
* @param searchKey a user defined input to find in the array
* @return element
* @throws MyException
*/
public int find(long searchKey) {
```

```
if(currentIndex == 0) {
System.out.println("Array is empty.");
}
else {
for(int i = 0; i < currentIndex; i++) {
if(a[i] == searchKey) {
return i;
}
}
}
return -1;
}
/**
* This method inserts value in the array at the end.
* @param value a user defined input which is to be inserted in the array
*/
public void insert(long value) {
a[currentIndex++] = value;
}
/*** This method returns the element at given index position.
* @param index position given by user
* @return element found at given index position
*/
public long getElem(int index) {
if(currentIndex == 0) {
System.out.println("Array is empty.");
return -1;
}
else if(index > a.length) {
System.out.println("Index out of bounds.");
return -1;
}
else {
return a[index];
}
}
/**
```

```
* This method deletes a single element from the array.
* @param value element to delete
* @return boolean value depending on if the element is deleted or not.
*/
public boolean delete(long value) {
if(currentIndex == 0) {
System.out.println("Array is empty.");
}
else {
for(int i = 0; i < currentIndex; i++) {
if(a[i] == value) {
for(int j = i; j < currentIndex - 1; j++) {
a[j] = a[j + 1];
}
currentIndex--;
return true;
}
}
}
return false;
}
/**
* This method displays all the elements of array.
*/
public void display() {
if(currentIndex == 0) {
System.out.println("Array is empty.");
}
else {
for(int i = 0; i < currentIndex; i++) {
System.out.println(a[i]);
}
}
}
/**
* This method deletes all the instance of given element.
* @param value element to delete
* @return count number of elements deleted*/
public int dupDelete(long value) {
int count = 0;
if(currentIndex == 0) {
System.out.println("Array is empty.");
return -1;
}
else {
for(int i = 0; i < currentIndex; i++) {
if(a[i] == value) {
delete(value);
count++;
}
}
return count;
}
}
/**
* This method overloads the insert() method and inserts the element at given
index position.
* @param index position at which to insert value
* @param value element to be inserted
*/
public void insert(int index, long value) {
if(getRemainingPositions() != 0) {
for(int i = currentIndex; i > index; i--) {
a[i] = a[i - 1];
}
a[index] = value;
currentIndex++;
System.out.println("Element inserted successfully.");
}
else {
System.out.println("Array is full.");
}
}
/**
```

```
* This methods deletes the element at given position.
* @param index position at which element is to be deleted.
* @return element which is deleted
*/
public long deleteAt(int index) {
if(currentIndex == 0) {
System.out.println("Array is empty.");
return -1;
}
else if(index > a.length) {
System.out.println("Array is full.");
return -1;
}
else {
long temp = a[index];
for(int i = index; i < currentIndex - 1; i++) {
a[i] = a[i + 1];
}
currentIndex--;
return temp;}
}
/**
* This method uses the java.util.Random.nextLong() method to initialize the array
with random long
values. If the array is full or not empty it prompts the user. The method only
initializes the array if it is
empty. The method also calls the display() method of the class to print all the
elements.
*/
public void initArray() {
Random rd = new Random();
int numofinitializedelements = 0;
if(getRemainingPositions() == 0) {
System.out.println("Array is full.");
}
else if(getRemainingPositions() < a.length) {
```

```
System.out.println("Array already has some elements in it, delete them first and
then initialize the array.");
}
else {
for(int i = 0; i < a.length; i++) {
a[currentIndex++] = rd.nextLong();
numofinitializedelements++;
}
System.out.println("Random elements entered are: " + numofinitializedelements);
System.out.println("Array elements are: ");
display();
}
}
/**
* This method sorts the array with the help of bubble sort. In bubble sort, zeroth
position element is
compared to all elements till n - 1 and if zeroth position element is greater than
any compared element then
we swap them and shift to right position. The number of comparisons in first
iteration is n - 1 and number of
swaps maybe between 0 to n - 1.
*/
public void bubbleSort() {
int in, out;
long temp;
for(out = a.length - 1; out > 1; out--) {
for(in = 0; in < out; in++) {
if(a[in] > a[in + 1]) {
temp = a[in];
a[in] = a[in + 1];
a[in + 1] = temp;
}
}
}
System.out.println("Array is sorted.");
}
/**
```

```
* This method sorts the array with the help of selection sort. In selection sort, we
first find the smallest
element in the array and swap it with the zeroth position element and shift right
by one position. The number
of swaps required is o(n) but, the number of comparisons is the same as bubble
sort which is o(n^2).
*/
public void selectionSort() {
int in, out, minIndex;
long temp;
for(out = 0; out < a.length; out++) {minIndex = out;
for(in = out + 1; in < a.length; in++) {
if(a[in] < a[minIndex]) {
minIndex = in;
}
}
temp = a[out];
a[out] = a[minIndex];
a[minIndex] = temp;
}
System.out.println("Array is sorted.");
}
/**
* This method uses insertion sort to sort the array elements. In insertion sort, a
marked item is chosen and
elements left to the marked item are considered partially sorted and the elements
right of the marked element
are considered unsorted. Then we find a place for marked element in the partially
sorted subarray and
remove the marked element from its place. Then we shift the elements greater
than marked element in the
sub array to the right and insert the marked element.
*/
public void insertionSort() {
int in, out;
long temp;
for(out = 1; out < a.length; out++) {
```

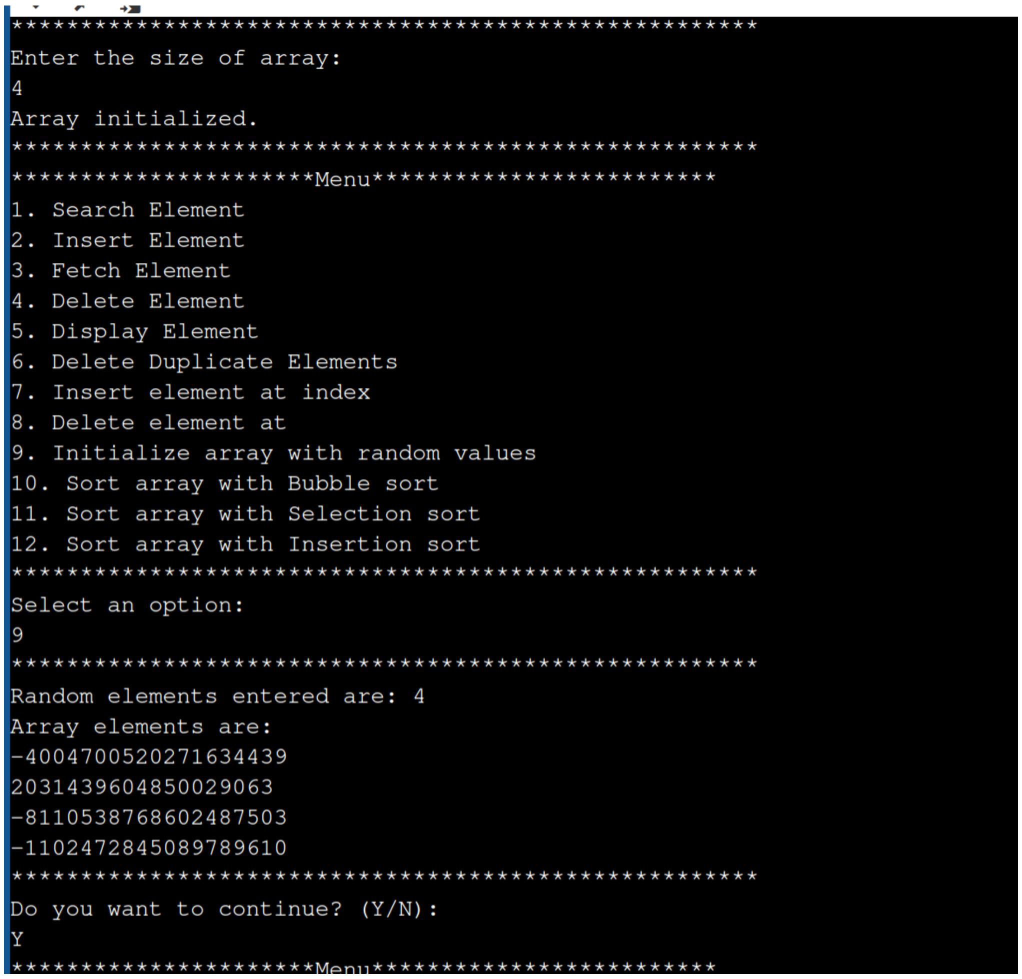
```
temp = a[out];
in = out;
while(in > 0 && a[in - 1] >= temp) {
a[in] = a[in - 1];
--in;
}
a[in] = temp;
}
System.out.println("Array is sorted.");
}
}
/**
* This class creates an object of MyLongArray Class and uses the various
methods of the class.
*/
class Exp3 {
/**
* Main Method initializes and uses the attributes and methods of MyLongArray
Class based on user input
which is implemented using switch case, scanner, etc. This method also
implements a do-while loop which
iterates based on user input.
*/
public static void main(String args[]) {
Scanner scan = new Scanner(System.in);
MyLongArray arr;
String menu_option;
String choice = "y";
System.out.println("****************************************************");
System.out.println("Enter the size of array: ");
try {
arr = new MyLongArray(Integer.parseInt(scan.nextLine()));
}
catch(Exception e) {
System.out.println("Enter a numerical value only.");arr = new
MyLongArray(Integer.parseInt(scan.nextLine()));
}
```

```
System.out.println("Array initialized.");
System.out.println("****************************************************");
do {
System.out.println("*********************Menu************************");
System.out.println("1. Search Element");
System.out.println("2. Insert Element");
System.out.println("3. Fetch Element");
System.out.println("4. Delete Element");
System.out.println("5. Display Element");
System.out.println("6. Delete Duplicate Elements");
System.out.println("7. Insert element at index");
System.out.println("8. Delete element at");
System.out.println("9. Initialize array with random values");
System.out.println("10. Sort array with Bubble sort");
System.out.println("11. Sort array with Selection sort");
System.out.println("12. Sort array with Insertion sort");
System.out.println("****************************************************");
System.out.println("Select an option: ");
menu_option = scan.nextLine();
switch (menu_option) {
case "1":
System.out.println("****************************************************");
System.out.println("Enter the element to search: ");
try {
int res = arr.find(Long.parseLong(scan.nextLine()));
if(res == -1) {
System.out.println("Element not found.");
}
else {
System.out.println("Element found at " + res + " position.");
}
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("****************************************************");
break;
case "2":
if(arr.getRemainingPositions() == 0) {
```

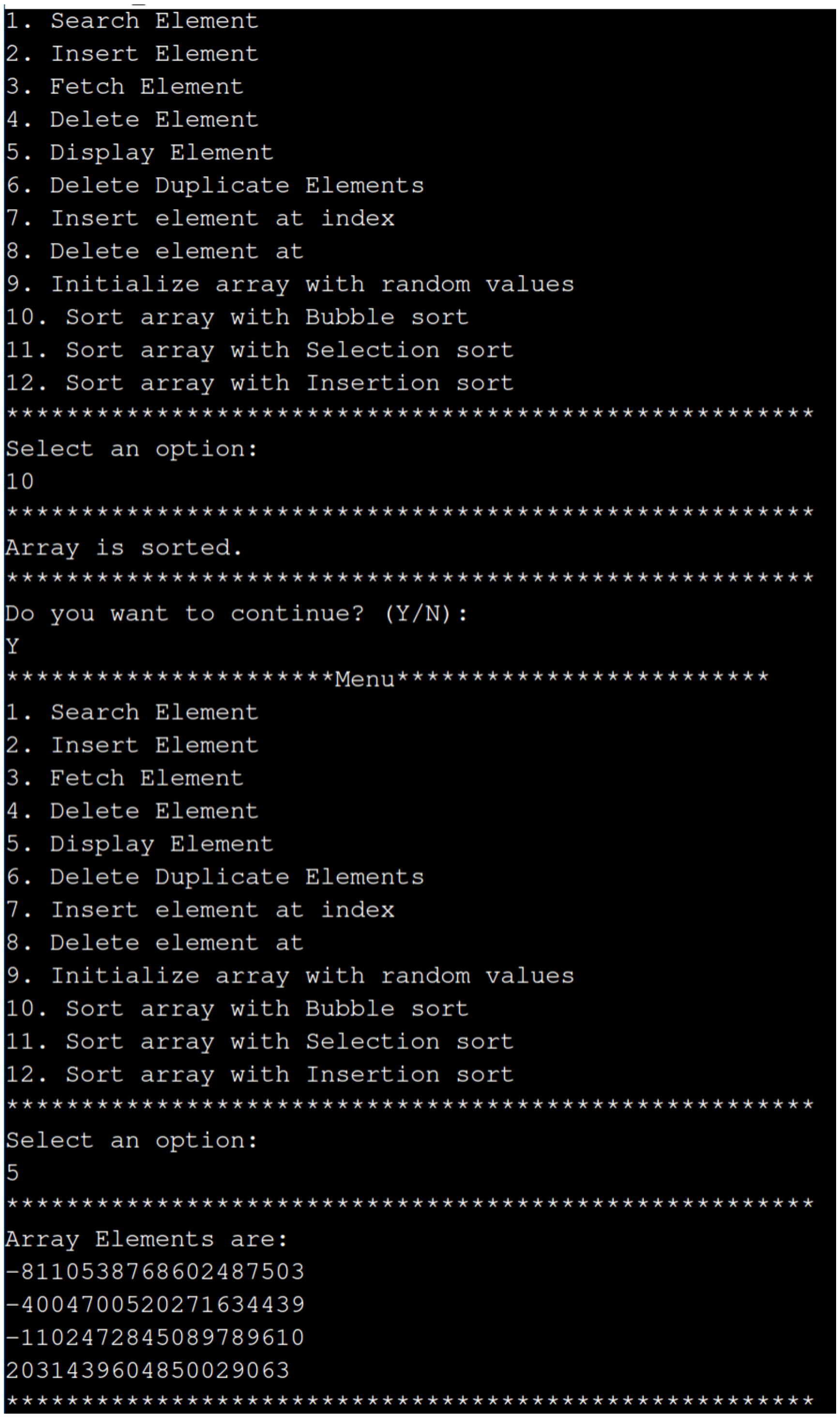
```
System.out.println("Array is full.");
}
else {
System.out.println("**********************************************************");
System.out.println("How many elements do you want to enter?: ");
try {
int counter = Integer.parseInt(scan.nextLine());
if(counter > arr.getRemainingPositions()) {
System.out.println("The entered value is greater than the available space in
array. Enter the number which is less than or equal to the space available(" +
arr.getRemainingPositions() + "): ");
counter = Integer.parseInt(scan.nextLine());
if(counter > arr.getRemainingPositions()) {
System.out.println("The entered value is greater than the available space in
array. Enter the number which is less than or equal to the space available(" +
arr.getRemainingPositions() + "): ");
break;
}}
for(int i = 1; i <= counter; i++) {
System.out.println("Enter the element to insert: ");
arr.insert(Long.parseLong(scan.nextLine()));
System.out.println("Inserted successfully.");
}
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("**********************************************************");
}
break;
case "3":
System.out.println("**********************************************************");
System.out.println("Enter the element to fetch: ");
try {
long elem = arr.getElem(Integer.parseInt(scan.nextLine()));
System.out.println("Element on the given index is: " + elem);
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("**********************************************************");
```

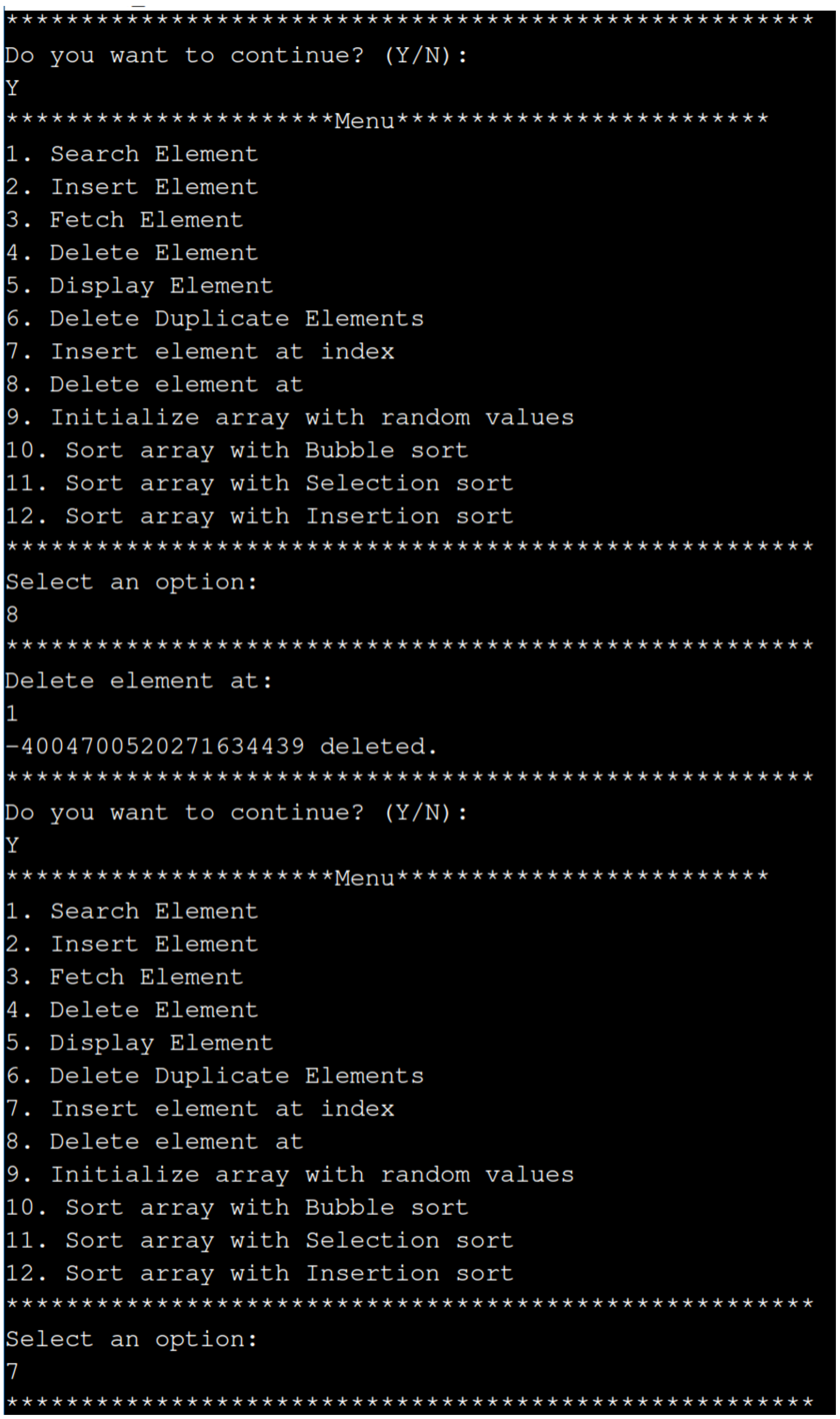
```
break;
case "4":
System.out.println("******************************************************");
System.out.println("Enter the element to delete: ");
try {
if(arr.delete(Long.parseLong(scan.nextLine()))) {
System.out.println("Element deleted.");
}
else {
System.out.println("Element not found.");
}
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("******************************************************");
break;
case "5":
System.out.println("******************************************************");
System.out.println("Array Elements are: ");
arr.display();
System.out.println("******************************************************");
break;
case "6":
System.out.println("******************************************************");
System.out.println("Enter the element to delete: ");
try {
int count = arr.dupDelete(Long.parseLong(scan.nextLine()));
System.out.println(count + " elements deleted.");
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("******************************************************");
break;case "7":
System.out.println("******************************************************");
System.out.println("Enter index and element to insert: (index element)");
try {
String str[] = scan.nextLine().split(" ");
int index = Integer.parseInt(str[0]);
long element = Long.parseLong(str[1]);
arr.insert(index, element);
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("*****************************************************");
break;
case "8":
System.out.println("*****************************************************");
System.out.println("Delete element at: ");
try {
long ele = arr.deleteAt(Integer.parseInt(scan.nextLine()));
System.out.println(ele + " deleted.");
}
catch(Exception e) {System.out.println("Enter a numerical value only.");}
System.out.println("*****************************************************");
break;
case "9":
System.out.println("*****************************************************");
arr.initArray();
System.out.println("*****************************************************");
break;
case "10":
System.out.println("*****************************************************");
arr.bubbleSort();
System.out.println("*****************************************************");
break;
case "11":
System.out.println("*****************************************************");
arr.selectionSort();
System.out.println("*****************************************************");
break;
case "12":
System.out.println("*****************************************************");
arr.insertionSort();
System.out.println("*****************************************************");
break;
default:
System.out.println("*****************************************************");
System.out.println("Wrong Choice!!");
System.out.println("*******************************************************");
break;
}
System.out.println("Do you want to continue? (Y/N): ");
choice = scan.nextLine();}while(choice.equals("y") || choice.equals("Yes") ||
choice.equals("Y") || choice.equals("yes"));
System.out.println("*******************************************************");
System.out.println("Thank you!!!");
scan.close();
}
}
```

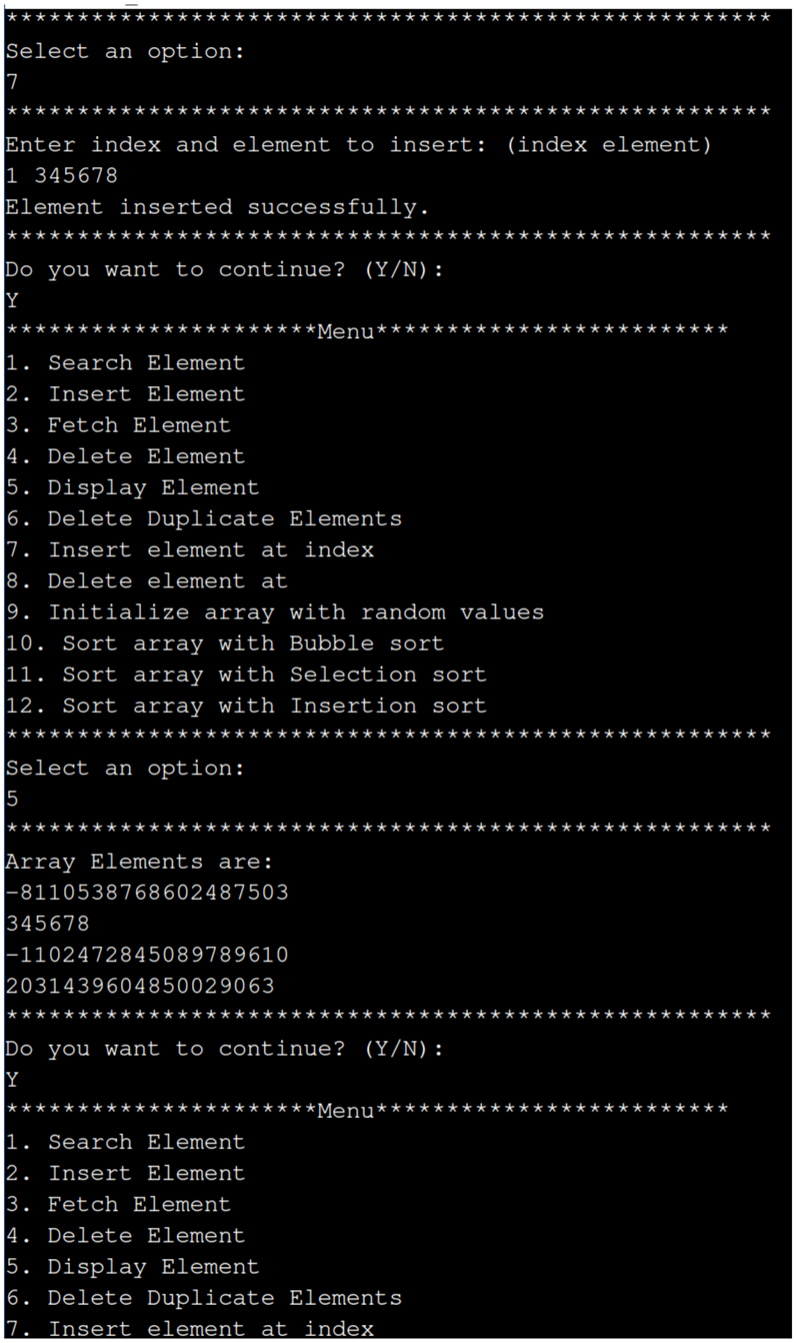
Output:



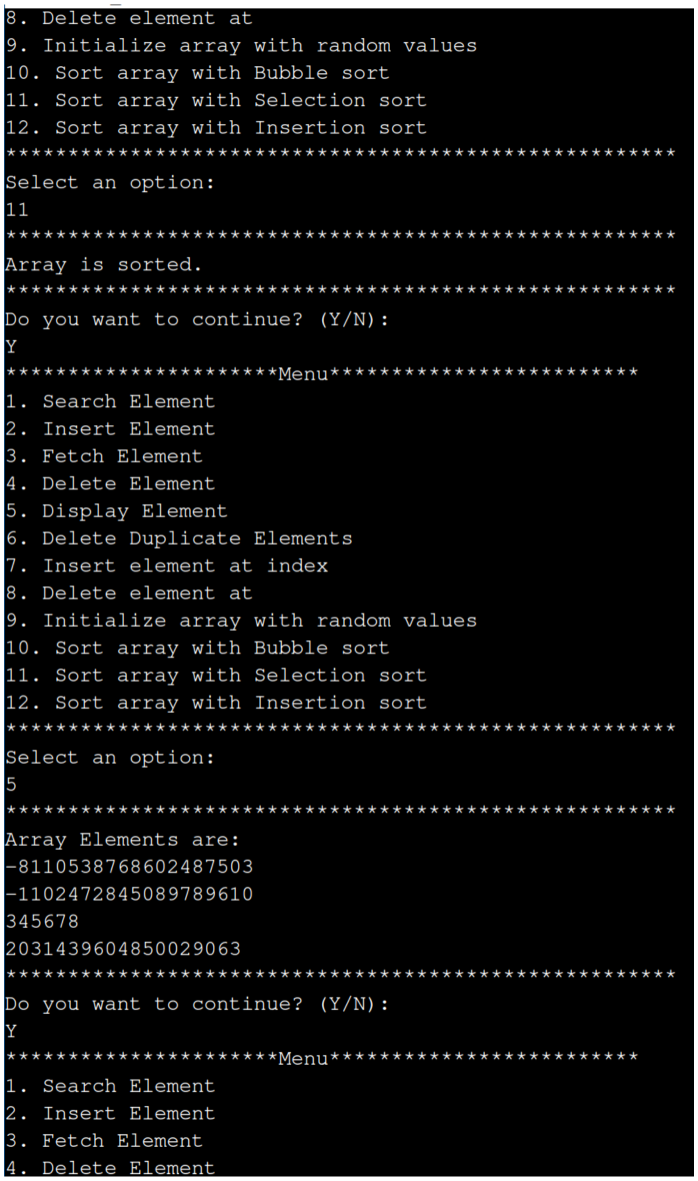
```
*******************************************************
Enter the size of array:
4
Array initialized.
*******************************************************
**********************Menu*************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*******************************************************
Select an option:
9
*******************************************************
Random elements entered are: 4
Array elements are:
-4004700520271634439
2031439604850029063
-8110538768602487503
-1102472845089789610
*******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu*************************
```


```
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
10
******************************************************
Array is sorted.
******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
5
******************************************************
Array Elements are:
-8110538768602487503
-4004700520271634439
-1102472845089789610
2031439604850029063
******************************************************
```

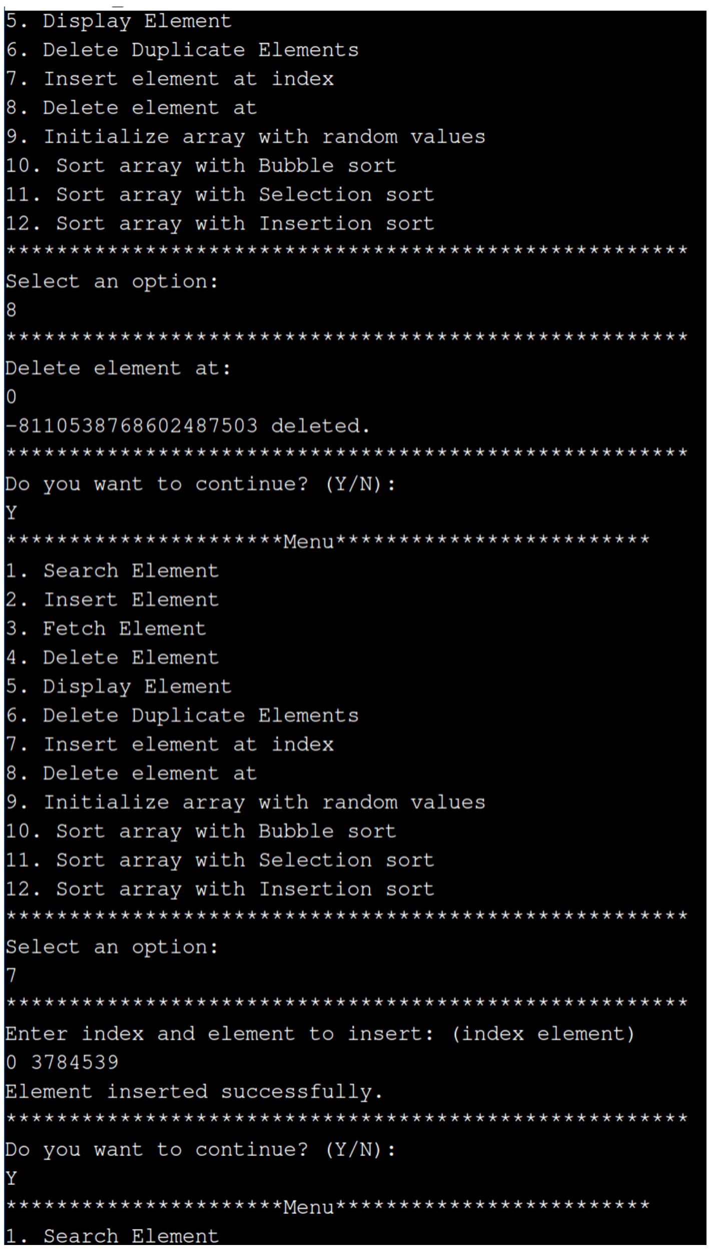

```
******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
8
******************************************************
Delete element at:
1
-4004700520271634439 deleted.
******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
7
******************************************************
```



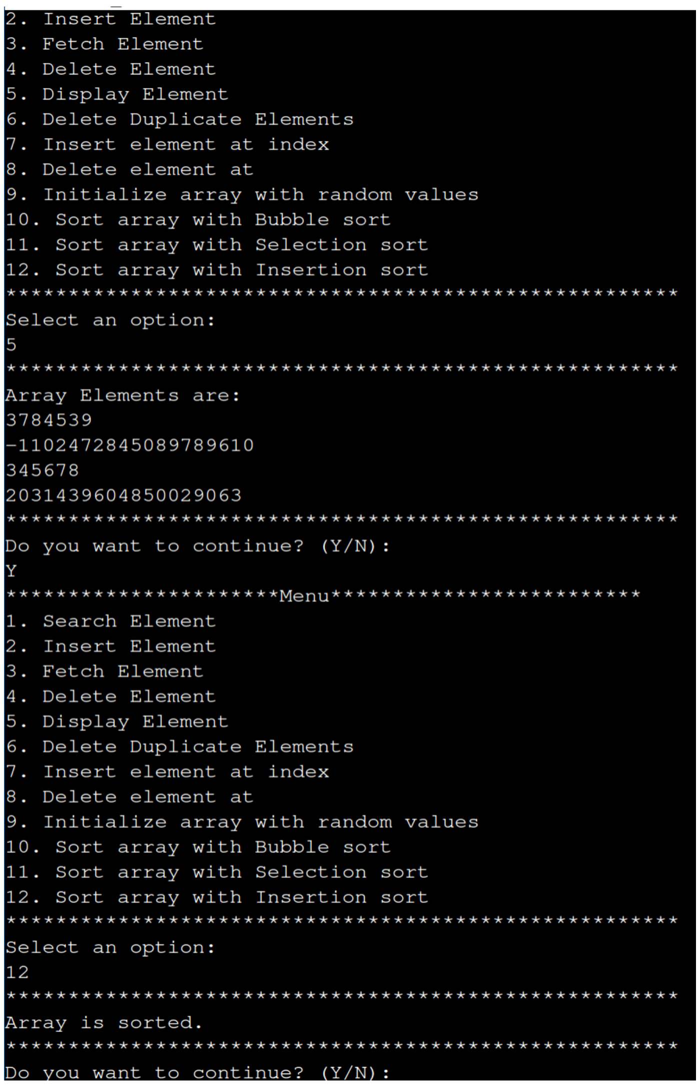
```
*****************************************************
Select an option:
7
*****************************************************
Enter index and element to insert: (index element)
1 345678
Element inserted successfully.
*****************************************************
Do you want to continue? (Y/N):
Y
*********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*****************************************************
Select an option:
5
*****************************************************
Array Elements are:
-8110538768602487503
345678
-1102472845089789610
2031439604850029063
*****************************************************
Do you want to continue? (Y/N):
Y
*********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
```



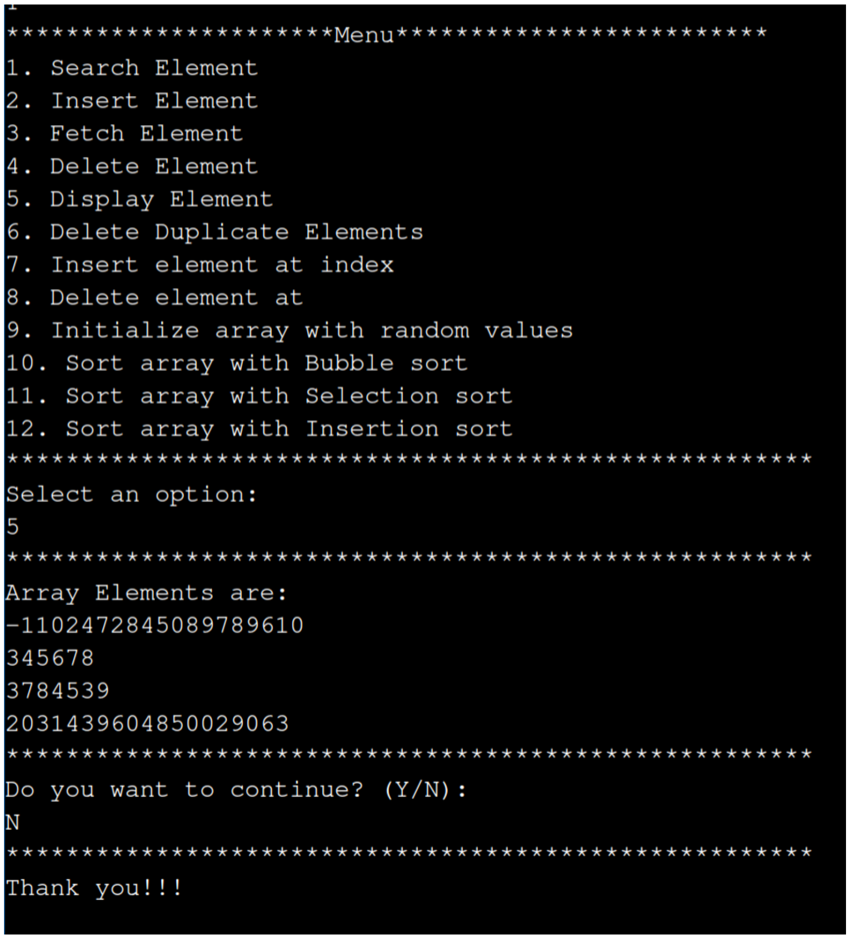
```
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*****************************************************
Select an option:
11
*****************************************************
Array is sorted.
*****************************************************
Do you want to continue? (Y/N):
Y
**********************Menu*************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*****************************************************
Select an option:
5
*****************************************************
Array Elements are:
-8110538768602487503
-1102472845089789610
345678
2031439604850029063
*****************************************************
Do you want to continue? (Y/N):
Y
**********************Menu*************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
```



```
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
8
******************************************************
Delete element at:
0
-8110538768602487503 deleted.
******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu*************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
******************************************************
Select an option:
7
******************************************************
Enter index and element to insert: (index element)
0 3784539
Element inserted successfully.
******************************************************
Do you want to continue? (Y/N):
Y
**********************Menu*************************
1. Search Element
```



```
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*****************************************************
Select an option:
5
*****************************************************
Array Elements are:
3784539
-1102472845089789610
345678
2031439604850029063
*****************************************************
Do you want to continue? (Y/N):
Y
*********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*****************************************************
Select an option:
12
*****************************************************
Array is sorted.
*****************************************************
Do you want to continue? (Y/N):
```



```
********************Menu************************
1. Search Element
2. Insert Element
3. Fetch Element
4. Delete Element
5. Display Element
6. Delete Duplicate Elements
7. Insert element at index
8. Delete element at
9. Initialize array with random values
10. Sort array with Bubble sort
11. Sort array with Selection sort
12. Sort array with Insertion sort
*************************************************
Select an option:
5
*************************************************
Array Elements are:
-1102472845089789610
345678
3784539
2031439604850029063
*************************************************
Do you want to continue? (Y/N):
N
*************************************************
Thank you!!!
```

Type of errors you encountered and the way you fixed it:
Error: Array Index Out of Bounds
Solution: The error occurred while traversing through the array while sorting it. The fix was just to give the
right condition in the loops to keep the array index from going outside the bounds.

Learning lessons from the experiment:

Through this experiment, I understood the need of sorting and how to implement sorting with three simple
sorting algorithms, mainly-
1. Bubble Sort: This algorithm is the slowest sorting algorithm but, it is also the simplest to understand and
implement. In Bubble sort, the leftmost element (which on position 0) is compared with every element

till n – 1, and if any element is smaller than the zeroth position-element we swap them and shift to right
position. The number of comparisons is o(n^2) and the number of swaps required are o(n^2).
2. Selection Sort: In selection sort, we first find and select the smallest element in the array and swap it with
zeroth position element and shift right by one position. While in bubble sort the sorted players are placed
on the right side, in selection sort the sorted players are placed in the left side or the start of the array. The
the number of comparisons required is o(n^2) and the number of swaps required is o(n).
3. Insertion Sort: In insertion sort, we first choose a marked element and consider the left-side elements of
the marked element to form a sub array which is partially sorted and right-side elements are considered
unsorted. Then we traverse through the sub array and find a place for the marked element. Then we remove
the marked element from its original position, move the elements of the sub array which are greater than
the marked element to the right and insert the marked element in the sub array. Continue this till all elements are sorted. Insertion sort's time complexity is o(n^2).
4.Learnt to use the `java.util.Random` class and how to call random datatypes using methods like `nextInt()`, `nextLong()`,

Conclusion:

To conclude, in this experiment we extended the `MyLongArray` class from assignment 02 with the methods:
'initArray()`, `bubbleSort()`, `selectionSort()`, `insertionSort()`. Through this assignment I understood the concept of sorting with the help of Bubble sort, Selection sort and Insertion sort and applied them in the code above.