EXPERIMENT –10 [Hashing]
Dated: 09.11.2023

1. Given a limited range array containing both positive and non-positive numbers, i.e., elements are in the range from -MAX to +MAX. Our task is to search if some number is present in the array or not in O(1) time.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX 10000

bool searchNumber(int hashTable[], int num) {

   if (hashTable[num + MAX] == 1) {
      return true;
   } else {
      return false;
   }
}
void insertNumber(int hashTable[], int num) {
   hashTable[num + MAX] = 1;
}

int main() {
   int hashTable[2 * MAX + 1] = {0};
   insertNumber(hashTable, -2);
   insertNumber(hashTable, 5);
   insertNumber(hashTable, 8);

   printf("%d\n", searchNumber(hashTable, -2)); // Should print 1 (true)
   printf("%d\n", searchNumber(hashTable, 0));  // Should print 0 (false)
   return 0;
}
```

Output:

```
1

0
```

2) Given two arrays: *A* and *B*. Find whether *B* is a subset of *A* or not using Hashing. Both the arrays are not in sorted order. It may be assumed that elements in both arrays are distinct.

Source code:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX 10000

bool isSubset(int A[], int m, int B[], int n) {
  int hashTable[MAX] = {0};

  for (int i = 0; i < m; i++) {
    hashTable[A[i]] = 1;
  }

  for (int i = 0; i < n; i++) {
    if (hashTable[B[i]] != 1) {
      return false;
    }
  }
  return true;
}

int main() {
  int A[] = {3, 7, 1, 9, 2};
  int B[] = {1, 9, 2};

  int sizeA = sizeof(A) / sizeof(A[0]);
  int sizeB = sizeof(B) / sizeof(B[0]);

  if (isSubset(A, sizeA, B, sizeB)) {
    printf("Array B is a subset of array A.\n");
  } else {
    printf("Array B is not a subset of array A.\n");
  }

  return 0;
}
```

OUTPUT:

Array B is a subset of array A.