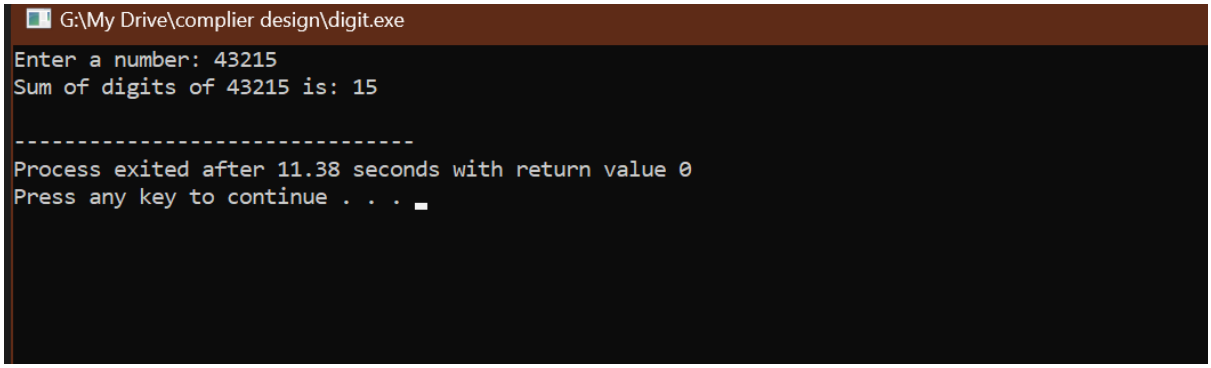
31, Write a program to find the sum of digits.

Code:

```
#include <stdio.h>
int sumOfDigits(int number) {
    if (number == 0) {
        return 0;
    } else {
        return (number % 10) + sumOfDigits(number / 10);
    }
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    int sum = sumOfDigits(num);
    printf("Sum of digits of %d is: %d\n", num, sum);
    return 0;
}
```



32, Write a program for to perform liner search.

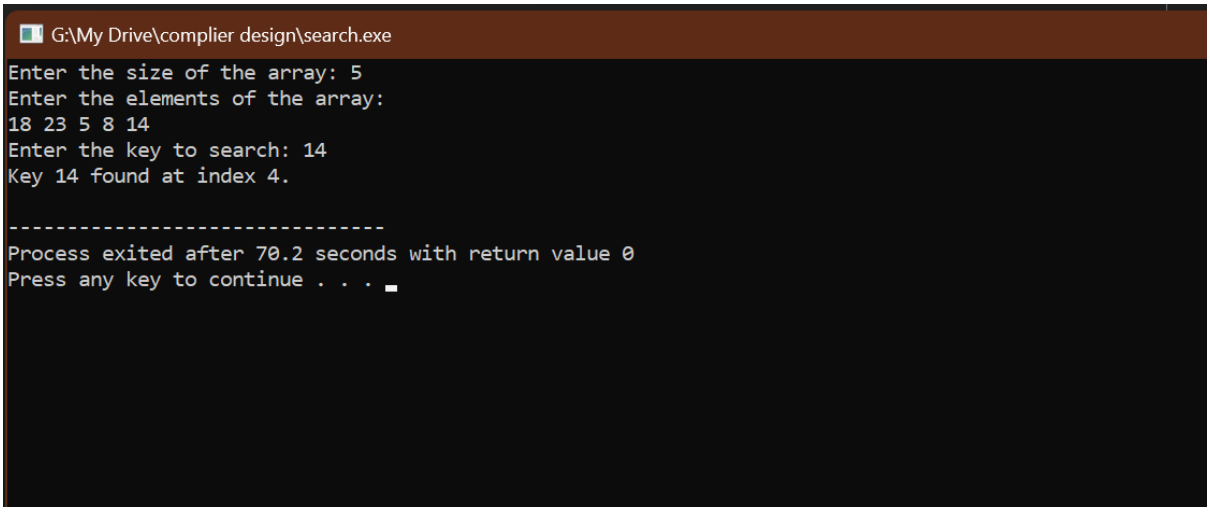
Code : #include <stdio.h>

```
int linearSearch(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
int main() {
    int size, key;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the key to search: ");
    scanf("%d", &key);
```

```
    int index = linearSearch(arr, size, key);
    if (index != -1) {
        printf("Key %d found at index %d.\n", key, index);
    } else {
        printf("Key %d not found in the array.\n", key);
    }

    return 0;
}
```

Output:



33, Write a program to perform n Queens problem using backtracking.

Code: #include <stdio.h>

#include <stdbool.h>

#define N 8

void printSolution(int board[N][N]) {

    for (int i = 0; i < N; i++) {

```
        for (int j = 0; j < N; j++) {
            printf("%2d ", board[i][j]);
        }
        printf("\n");
    }
}
bool isSafe(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;
    for (int i = row, j = col; i < N && j >= 0; i++, j--)
        if (board[i][j])
            return false;

    return true;
}
bool solveNQueensUtil(int board[N][N], int col) {
    if (col >= N)
        return true;

    for (int i = 0; i < N; i++) {
```

```
        if (isSafe(board, i, col)) {
            board[i][col] = 1;

            if (solveNQueensUtil(board, col + 1))
                return true;

            board[i][col] = 0;
        }
    }

    return false;
}
void solveNQueens() {
    int board[N][N] = {{0}};

    if (!solveNQueensUtil(board, 0))
        printf("Solution does not exist");
    else
        printSolution(board);
}

int main() {
    solveNQueens();
```

```
    return 0;
}
```

Output:

34, Write a program to inset a number in a list.

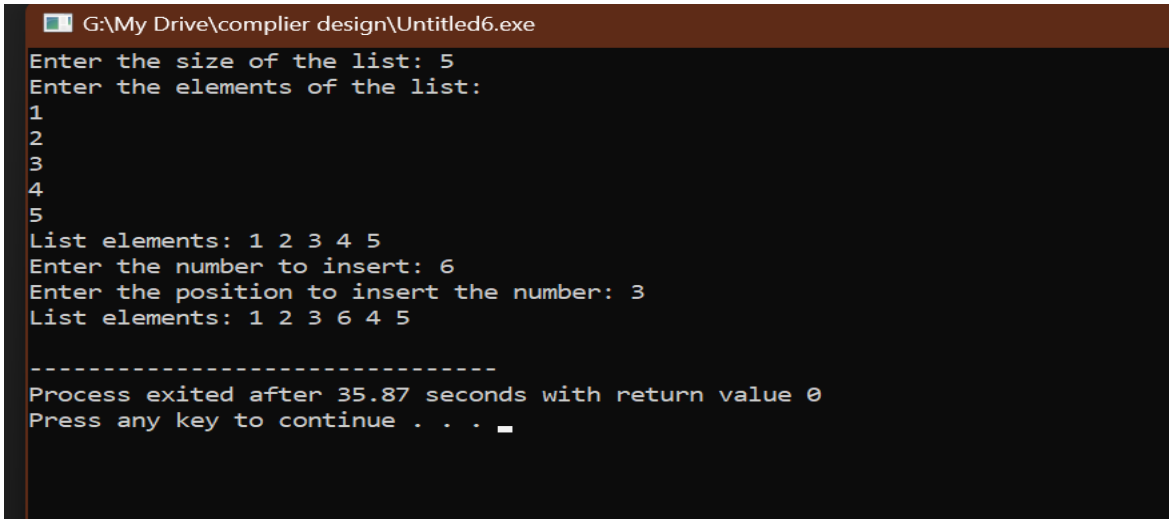
Code:

```
#include <stdio.h>
void displayList(int list[], int size) {
    printf("List elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", list[i]);
    }
    printf("\n");
```

```
}
void insertNumber(int list[], int *size, int number, int position) {
    if (position < 0 || position > *size) {
        printf("Invalid position. Please enter a valid position.\n");
        return;
    }
    for (int i = *size; i > position; i--) {
        list[i] = list[i - 1];
    }
    list[position] = number;
    (*size)++;
}
int main() {
    int list[50];
    int size, number, position;
    printf("Enter the size of the list: ");
    scanf("%d", &size);
    printf("Enter the elements of the list:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &list[i]);
    }
    displayList(list, size);
    printf("Enter the number to insert: ");
    scanf("%d", &number);
```

```
    printf("Enter the position to insert the number: ");
    scanf("%d", &position);
    insertNumber(list, &size, number, position);
    displayList(list, size);
    return 0;
}
```

Output:



35, Write a program to perform sum of subsets problem using backtracking.

Code:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
void printSubset(int subset[], int size) {
    printf("Subset with the given sum: { ");
    for (int i = 0; i < size; i++) {
```

```
        printf("%d ", subset[i]);
    }
    printf("}\n");
}
bool isSubsetSum(int set[], int n, int sum, int subset[], int subsetSize) {
    if (sum == 0) {
        printSubset(subset, subsetSize);
        return true;
    }
    if (n == 0 && sum != 0)
        return false;
    if (set[n - 1] > sum)
        return isSubsetSum(set, n - 1, sum, subset, subsetSize);
    return isSubsetSum(set, n - 1, sum, subset, subsetSize) ||
           isSubsetSum(set, n - 1, sum - set[n - 1], subset, subsetSize + 1);
}
void subsetSum(int set[], int n, int sum) {
    int subset[MAX_SIZE];
    if (!isSubsetSum(set, n, sum, subset, 0))
        printf("No subset with the given sum exists.\n");
}
int main() {
    int set[] = {3, 34, 4, 12, 5, 2};
    int n = sizeof(set) / sizeof(set[0]);
```

```
    int targetSum = 9;

    subsetSum(set, n, targetSum);

    return 0;

}
```

Output: