`1. Flight Ticket Booking System`
`Scenario:`
A travel agency wants to `store the details of N passengers` (Name, Age, Destination).
The system should allow:
- Adding a `new passenger`
- `Sorting` passengers by `destination name (A-Z)`
- Searching for passengers traveling to a specific destination

Use `arrays` and implement `sorting and searching techniques`.

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

#define MAX_PASSENGERS 100
#define MAX_NAME_LENGTH 50
#define MAX_DESTINATION_LENGTH 50

typedef struct {
    char name[MAX_NAME_LENGTH];
    int age;
    char destination[MAX_DESTINATION_LENGTH];
} Passenger;

void addPassenger(Passenger passengers[], int *count) {
    if (*count >= MAX_PASSENGERS) {
        printf("Cannot add more passengers. Maximum capacity reached.\n");
        return;
    }

    printf("Enter passenger %d details (Name, Age, Destination): ", *count
+ 1);
    scanf("%s %d %s", passengers[*count].name, &passengers[*count].age,
passengers[*count].destination);
    (*count)++;
}

void sortPassengers(Passenger passengers[], int count) {
    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (strcmp(passengers[j].destination, passengers[j +
1].destination) > 0) {
```

```
                // Swap passengers
                Passenger temp = passengers[j];
                passengers[j] = passengers[j + 1];
                passengers[j + 1] = temp;
            }
        }
    }
}

void searchPassengers(Passenger passengers[], int count, const char
*destination) {
    int found = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(passengers[i].destination, destination) == 0) {
            printf("%s - %d - %s\n", passengers[i].name,
passengers[i].age, passengers[i].destination);
            found = 1;
        }
    }
    if (!found) {
        printf("No passengers found traveling to %s.\n", destination);
    }
}

int main() {
    Passenger passengers[MAX_PASSENGERS];
    int count = 0;
    int n;

    printf("Enter number of passengers: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        addPassenger(passengers, &count);
    }

    sortPassengers(passengers, count);

    printf("\nSorted List (by destination):\n");
    for (int i = 0; i < count; i++) {
        printf("%s - %s\n", passengers[i].name,
passengers[i].destination);
    }

    char searchDestination[MAX_DESTINATION_LENGTH];
    printf("\nEnter destination to search: ");
    scanf("%s", searchDestination);

    printf("Passengers traveling to %s:\n", searchDestination);
    searchPassengers(passengers, count, searchDestination);

    return 0;
}
```

```
Enter number of passengers: 3
Enter passenger 1 details (Name, Age, Destination): Anne
15
Mumbai
Enter passenger 2 details (Name, Age, Destination): Rohit
19
Delhi
Enter passenger 3 details (Name, Age, Destination): Ryan
20
Chennai

Sorted List (by destination):
Ryan - Chennai
Rohit - Delhi
Anne - Mumbai

Enter destination to search: Chennai
Passengers traveling to Chennai:
Ryan - 20 - Chennai
```

`2. DNA Sequence Pattern Finder`
`Scenario:`
A `biotech lab` is analyzing DNA sequences. Given a `DNA string` (containing only 'A', 'T', 'G', 'C'), check if a `specific pattern exists in it`.

Use `string functions` to `find and count occurrences` of a given pattern.

`Input Example:`

```
Enter DNA Sequence: ATGCGATCGT
Enter pattern to search: ATG
```

`Output Example:`
```
Pattern found 1 time(s) in the DNA sequence.
```

```c
#include <stdio.h>
#include <string.h>

#define MAX_SEQUENCE_LENGTH 1000

int countPatternOccurrences(const char *dna, const char *pattern) {
    int count = 0;
    int patternLength = strlen(pattern);
    int dnaLength = strlen(dna);

    for (int i = 0; i <= dnaLength - patternLength; i++) {
        if (strncmp(&dna[i], pattern, patternLength) == 0) {
            count++;
        }
    }

    return count;
}

int main() {
    char dna[MAX_SEQUENCE_LENGTH];
    char pattern[MAX_SEQUENCE_LENGTH];

    printf("Enter DNA Sequence: ");
    scanf("%s", dna);

    printf("Enter pattern to search: ");
    scanf("%s", pattern);

    int occurrences = countPatternOccurrences(dna, pattern);
    printf("Pattern found %d time(s) in the DNA sequence.\n",
occurrences);

    return 0;
}
```

Enter DNA Sequence: atcgtgacag
Enter pattern to search: gtg
Pattern found 1 time(s) in the DNA sequence.

`3. Cricket Scoreboard System`
`Scenario:`
A `cricket club` records match scores in an array.
The system should:
- Find the `highest and lowest score`
- Calculate the `average score`

Use `arrays` for data storage and `looping techniques` for computation.

`Input Example:`
```
Enter scores of 5 matches: 245 189 320 270 150
```

`Output Example:`
```
Highest Score: 320
Lowest Score: 150
Average Score: 234.8
```

```
#include <stdio.h>

#define MAX_MATCHES 100

void findHighestLowest(int scores[], int n, int *highest, int *lowest) {
    *highest = scores[0];
    *lowest = scores[0];

    for (int i = 1; i < n; i++) {
        if (scores[i] > *highest) {
            *highest = scores[i];
        }
```

```c
        if (scores[i] < *lowest) {
            *lowest = scores[i];
        }
    }
}

float calculateAverage(int scores[], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += scores[i];
    }
    return (float)sum / n;
}

int main() {
    int scores[MAX_MATCHES];
    int n;

    printf("Enter number of matches: ");
    scanf("%d", &n);

    printf("Enter scores of %d matches: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &scores[i]);
    }

    int highest, lowest;
    findHighestLowest(scores, n, &highest, &lowest);
    float average = calculateAverage(scores, n);

    printf("Highest Score: %d\n", highest);
    printf("Lowest Score: %d\n", lowest);
    printf("Average Score: %.1f\n", average);

    return 0;
}
```

```
Enter number of matches: 5
Enter scores of 5 matches: 450
45
```

```
513
258
97
Highest Score: 513
Lowest Score: 45
Average Score: 272.6
```