SOURCE CODE:

```
#include <stdio.h>
#define MAX_PARKING_SLOTS 100

int main() {
  int
  parking[MAX_PARKING_S
  LOTS]; int totalSlots,
  carCount;
  int carNumber, slotPosition;

  // Step 1: Input total number of parking slots and cars parked
  printf("Enter total number of parking slots (max %d): ",
  MAX_PARKING_SLOTS); scanf("%d", &totalSlots);

  if (totalSlots <= 0 || totalSlots >
    MAX_PARKING_SLOTS) { printf("Invalid slot
    number.\n");
    return 1;
  }


  printf("Enter number of cars currently
  parked: "); scanf("%d", &carCount);

  if (carCount < 0 || carCount > totalSlots)
    { printf("Invalid car count.\n");
    return 1;
  }

  // Step 2: Input car numbers for parked cars
  printf("Enter car numbers (only %d
  values):\n", carCount); for (int i = 0; i <
  carCount; i++) {
    scanf("%d", &parking[i]);
  }


  // Mark remaining slots as empty (-1)
  for (int i = carCount; i < totalSlots;
    i++) { parking[i] = -1;
```

```
    }

    // Step 3: Display current
    parking lot printf("Current
    parking slots:\n"); for (int i
    = 0; i < totalSlots; i++) {
      if (parking[i] == -1)
        printf("Slot %d:
      [Empty]\n", i); else
        printf("Slot %d: Car %d\n", i, parking[i]);
    }

    // Step 4: Insert new car
    printf("Enter new car number to
    park: "); scanf("%d",
    &carNumber);
    printf("Enter slot number to park the car (0 to %d): ",
    totalSlots - 1); scanf("%d", &slotPosition);

    if (slotPosition < 0 || slotPosition >= totalSlots) {

  printf("Invalid slot
  number.\n"); return 1;
    }

    if (parking[slotPosition] != -1) {
      printf("Slot already occupied by Car %d.\n",
      parking[slotPosition]); return 1;
    }
    // Step 5: Park the new car
    parking[slotPosition] =
    carNumber;

    // Step 6: Display updated
    parking printf("\nUpdated
    parking slots:\n"); for (int i =
    0; i < totalSlots; i++) {
      if (parking[i] == -1)
        printf("Slot %d:
      [Empty]\n", i); else
```

```
        printf("Slot %d: Car %d\n", i, parking[i]);
    }

    return 0;
}
```

SAMPLE INPUT:

Enter total number of parking slots (max 100): 5
Enter number of cars currently parked: 3

Enter car numbers (only 3
values): 101 102 103

Enter new car number to park: 200

Enter slot number to park the car (0 to 4): 4

SAMPLE OUTPUT:

Current parking slots:

Slot 0: Car 101

Slot 1: Car 102

Slot 2: Car 103

Slot 3: [Empty]

Slot 4: [Empty]

Enter new car number to park: 200

Enter slot number to park the car (0 to 4): 4

Updated parking
slots:

Slot 0: Car 101

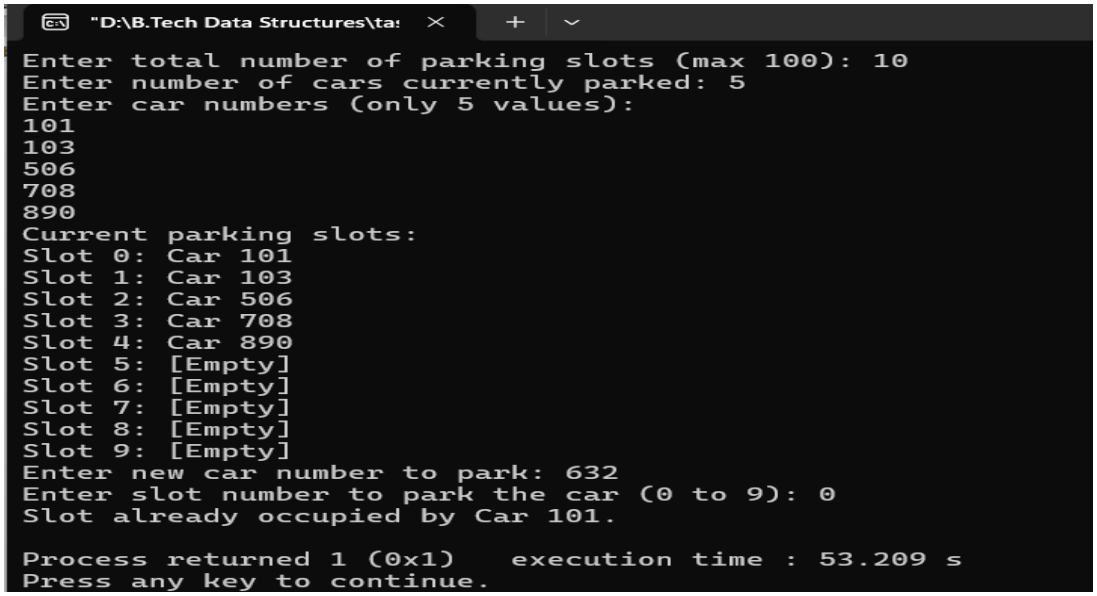
Slot 1: Car 102

Slot 2: Car 103

Slot 3: [Empty]

Slot 4: Car 200

OUTPUT :


"D:\B.Tech Data Structures\ta
Enter total number of parking slots (max 100): 10
Enter number of cars currently parked: 5
Enter car numbers (only 5 values):
101
103
506
708
890
Current parking slots:
Slot 0: Car 101
Slot 1: Car 103
Slot 2: Car 506
Slot 3: Car 708
Slot 4: Car 890
Slot 5: [Empty]
Slot 6: [Empty]
Slot 7: [Empty]
Slot 8: [Empty]
Slot 9: [Empty]
Enter new car number to park: 632
Enter slot number to park the car (0 to 9): 0
Slot already occupied by Car 101.
Process returned 1 (0x1) execution time : 53.209 s
Press any key to continue.

SOURCE CODE:

```
#include <stdio.h>

#define MAX

100 int main()

{
  int
  vtuno[MAX]
  ; int n, i, j;
  int foundDuplicate = 0;

  // Step 1: Input number of students
  printf("Enter number of students (max
  %d): ", MAX); scanf("%d", &n);

  if (n <= 0 || n > MAX) {
    printf("Invalid number of
    students.\n"); return 1;
  }


  // Step 2: Input VTUNO IDs
  printf("Enter VTUNO (Student
  IDs):\n"); for (i = 0; i < n; i++)
  {
    scanf("%d", &vtuno[i]);
  }

  // Step 3: Check for duplicates using
  nested loop printf("Duplicate
  VTUNO(s):\n");
  for (i = 0; i < n; i++) {
    for (j = i + 1; j < n; j++) {
      if (vtuno[i] == vtuno[j]) {
        printf("%d\n", vtuno[i]);
        foundDuplicate = 1;
        break; // Print only once per duplicate
      }
```

```
    }
  }

  if (!foundDuplicate) {
    printf("No duplicates found.\n");
  }

  return 0;
}
```

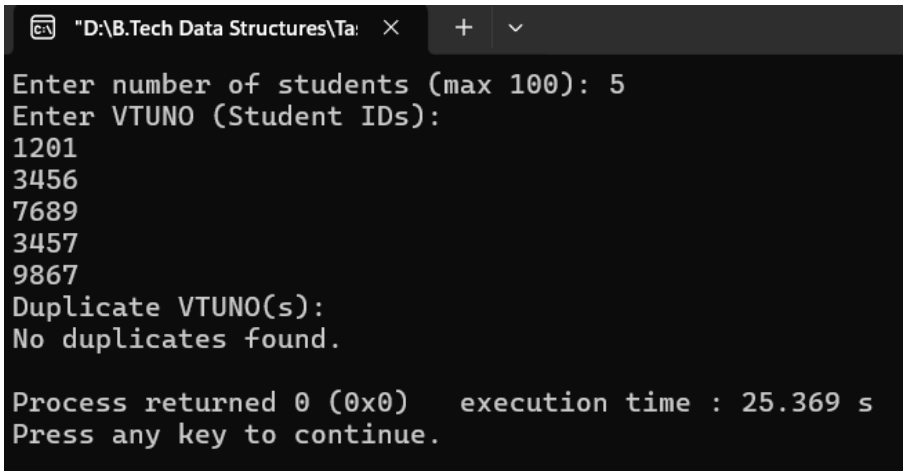
SAMPLE INPUT:

```
Enter number of students (max 100): 5
Enter VTUNO (Student IDs):
101 102 103 102 104
```

SAMPLE OUTPUT:

```
Enter number of students (max 100): 4
Enter VTUNO (Student IDs):
201 202 203 204
No duplicates found.
```

OUTPUT :


"D:\B.Tech Data Structures\Ta
Enter number of students (max 100): 5
Enter VTUNO (Student IDs):
1201
3456
7689
3457
9867
Duplicate VTUNO(s):
No duplicates found.
Process returned 0 (0x0) execution time : 25.369 s
Press any key to continue.

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int n, start, end;

    printf("Enter number of patients with data: ");
    scanf("%d", &n);

    int *patientID = (int*)malloc(n * sizeof(int));
    printf("Enter %d patient IDs:\n", n);
    for(int i = 0; i < n; i++) {
        scanf("%d", &patientID[i]);
    }

    printf("Enter start and end of patient ID range:\n");
    scanf("%d %d", &start, &end);
    int range = end - start + 1;

    int *present = (int*)calloc(range, sizeof(int));
    for(int i = 0; i < n; i++) {
        if(patientID[i] >= start && patientID[i] <= end) {
            present[patientID[i] - start] = 1;
```

```
        }
    }

    int missingFound = 0;
    printf("Missing patient IDs in range %d to %d:\n", start, end);
    for(int i = 0; i < range; i++) {
        if(present[i] == 0) {
            printf("%d ", i + start);
            missingFound = 1;
        }
    }

    if(!missingFound) {
        printf("-1");
    }

    printf("\n");

    free(patientID);
    free(present);
    return 0;
}
```

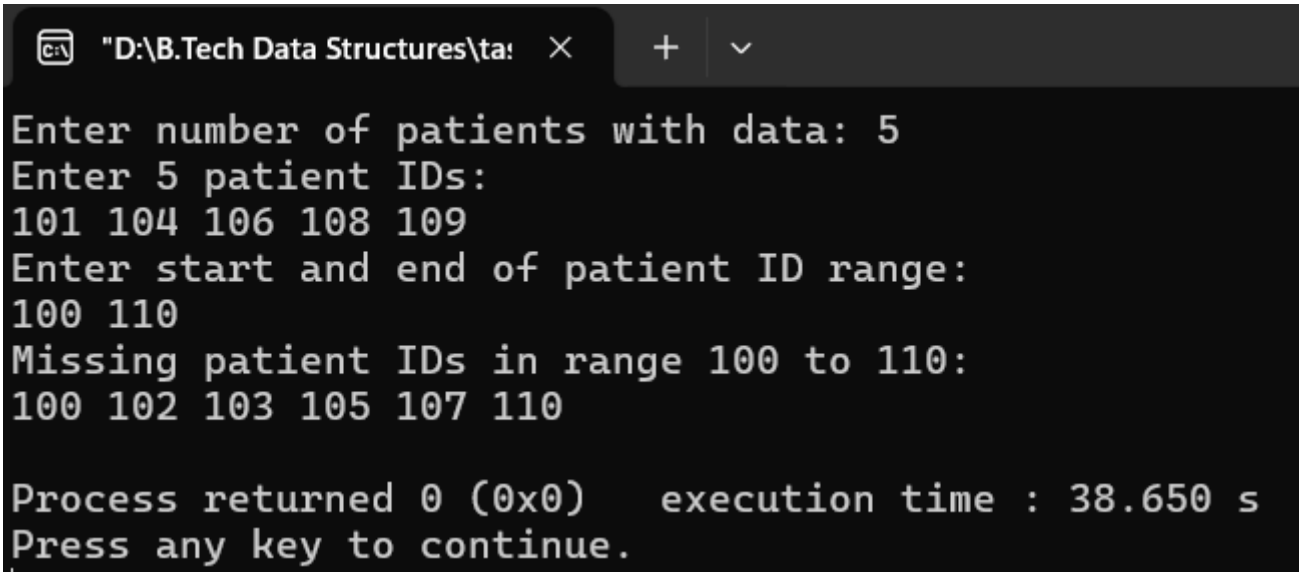
SAMPLE INPUT:

Enter number of patients with data: 5
Enter 5 patient IDs:
101 103 105 106 108
Enter start and end of patient ID range:
101 108

SAMPLE OUTPUT:

Missing patient IDs in range 101 to 108:
102 104 107

OUTPUT :


"D:\B.Tech Data Structures\ta
Enter number of patients with data: 5
Enter 5 patient IDs:
101 104 106 108 109
Enter start and end of patient ID range:
100 110
Missing patient IDs in range 100 to 110:
100 102 103 105 107 110
Process returned 0 (0x0) execution time : 38.650 s
Press any key to continue.

SOURCE CODE:

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

// Function to find minimum absolute difference using brute force
int minDiffBruteForce(int arr[], int n) {
    int minDiff = INT_MAX;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            int absDiff = abs(arr[i] - arr[j]);
            if (absDiff < minDiff) {
                minDiff = absDiff;
            }
        }
    }
    return minDiff;
}

int main() {
    int n;
    printf("Enter number of temperature readings: ");
    scanf("%d", &n);
```

```
    int a[n];
    printf("Enter %d temperature readings:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    printf("Minimum Difference is %d\n", minDiffBruteForce(a, n));
    return 0;
}
```

SAMPLE INPUT:

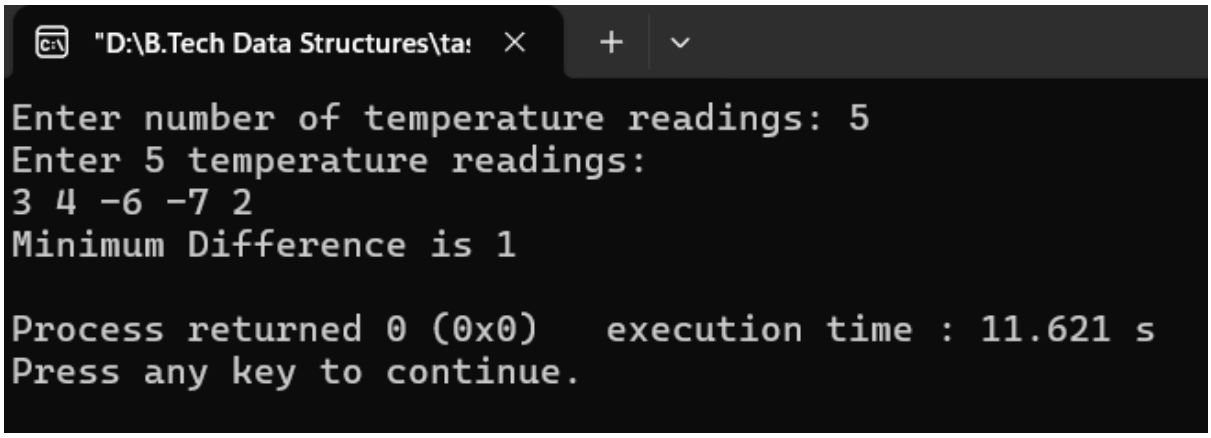
Enter number of temperature readings: 5 Enter 5 temperature readings:

3 4 -6 -7 2

SAMPLE OUTPUT:

Minimum Difference is 1

OUTPUT :


"D:\B.Tech Data Structures\ta:
Enter number of temperature readings: 5
Enter 5 temperature readings:
3 4 -6 -7 2
Minimum Difference is 1
Process returned 0 (0x0)   execution time : 11.621 s
Press any key to continue.

SOURCE CODE:

```
#include <stdio.h>
int main() {
    int n, k;

    // Input the number of songs in the playlist
    printf("Enter the number of songs in the playlist: ");
    scanf("%d", &n);

    int playlist[n];
    // Input the song IDs
    printf("Enter the song IDs in the playlist:\n");
    for(int i = 0; i < n; i++) {
        scanf("%d", &playlist[i]);
    }

    // Input number of rotation steps
    printf("Enter the number of steps to rotate: ");
    scanf("%d", &k);

    // Handle rotations greater than playlist size
    k = k % n;
```

```
// Display original playlist
printf("Original playlist:\n");
for(int i = 0; i < n; i++) {
   printf("%d ", playlist[i]);
}
printf("\n");

// Perform circular rotation
for(int i = 0; i < k; i++) {
   int temp = playlist[n - 1]; // store last element
   for(int j = n - 1; j > 0; j--) {
      playlist[j] = playlist[j - 1]; // shift elements right
   }
   playlist[0] = temp; // place last element at the start
}

// Display playlist after rotation
printf("Playlist after rotation:\n");
for(int i = 0; i < n; i++) {
   printf("%d ", playlist[i]);
}
printf("\n");
```

```
    return 0;
}
```

SAMPLE INPUT:

Enter the number of songs in the playlist: 5 Enter the song IDs in the playlist:

101 102 103 104 105

Enter the number of steps to rotate: 2 SAMPLE OUTPUT:

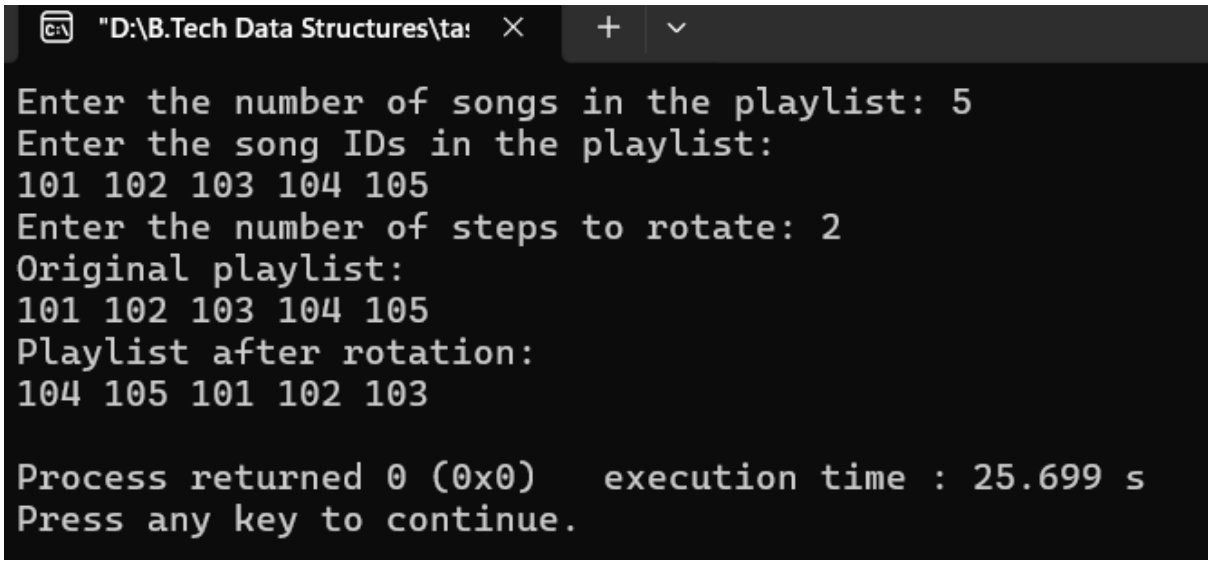
Original playlist:

101 102 103 104 105

Playlist after rotation:

104 105 101 102 103

OUTPUT :


"D:\B.Tech Data Structures\ta
Enter the number of songs in the playlist: 5
Enter the song IDs in the playlist:
101 102 103 104 105
Enter the number of steps to rotate: 2
Original playlist:
101 102 103 104 105
Playlist after rotation:
104 105 101 102 103
Process returned 0 (0x0) execution time : 25.699 s
Press any key to continue.