Problem: Write a function to remove all duplicates from an array without using additional data structures.

Topic: Arrays

```
def remove_duplicates(arr):
    arr.sort()  # Sort the array to group duplicates together
    index = 0
    for i in range(1, len(arr)):
        if arr[i] != arr[index]:
            index += 1
            arr[index] = arr[i]
    return arr[:index + 1]

# Example usage
print(remove_duplicates([4, 2, 4, 5, 2, 3, 1]))
```

#Output

```
[1, 2, 3, 4, 5]
```

Problem: Implement a binary search function to find the square root of a given number up to a certain decimal precision,

Topic: Searching Algorithms

```
def square_root(num, precision=0.0001):
    low, high = 0, num
    while high - low > precision:
        mid = (low + high) / 2
        if mid * mid < num:
            low = mid
        else:
            high = mid
    return round((low + high) / 2, int(-precision.log10()))

# Example usage
print(square_root(25))
print(square_root(10, 0.001))
```

#Output

```
5.0
3.162
```

Problem: Create a program to implement a circular queue using an array. Include enqueue, dequeue, and peek operations.

Topic: Queues

```
class CircularQueue:
    def __init__(self, size):
        self.size = size
        self.queue = [None] * size
        self.front = -1
        self.rear = -1

    def enqueue(self, value):
        if (self.rear + 1) % self.size == self.front:
            print("Queue is full")
            return
        if self.front == -1:
            self.front = 0
        self.rear = (self.rear + 1) % self.size
        self.queue[self.rear] = value

    def dequeue(self):
        if self.front == -1:
            print("Queue is empty")
            return None
        value = self.queue[self.front]
        if self.front == self.rear:
            self.front = self.rear = -1
        else:
            self.front = (self.front + 1) % self.size
        return value
    def peek(self):
        if self.front == -1:
            print("Queue is empty")
            return None
        return self.queue[self.front]

# Example usage
cq = CircularQueue(5)
cq.enqueue(1)
cq.enqueue(2)
cq.enqueue(3)
print(cq.dequeue())
print(cq.peek())
cq.enqueue(4)
cq.enqueue(5)
cq.enqueue(6)
```

#output

```
1
2
Queue is full
```