Lab:10.

Q1: Create an iterator that iterates through a list and returns only the unique elements. Test it with a list containing duplicates.

Program:

```
class UniqueIterator:
    def __init__(self, iterable):
        self.iterable = iterable
        self.visited = set()
        self.index = 0

    def __iter__(self):
        return self

    def __next__(self):
        while self.index < len(self.iterable):
            element = self.iterable[self.index]
            self.index += 1
            if element not in self.visited:
                self.visited.add(element)
                return element
        raise StopIteration
if __name__ == "__main__":
    input_list = input("Enter a list of elements separated by spaces: ").split()
        unique_iterator = UniqueIterator(input_list)
    print("Unique Elements:")
    for unique_element in unique_iterator:
```

```
        print(unique_element)
```

Q2: Implement an iterator that breaks a list into chunks of a specified size. Test it with a list of numbers and a chunk size of 3

Program:

```
class ChunkIterator:
    def __init__(self, iterable, chunk_size):
        self.iterable = iterable
        self.chunk_size = chunk_size
        self.index = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.index >= len(self.iterable):
            raise StopIteration

        chunk = self.iterable[self.index:self.index + self.chunk_size]
        self.index += self.chunk_size
        return chunk
if __name__ == "__main__":
    input_list = input("Enter a list of numbers separated by spaces: ").split()
    chunk_size = int(input("Enter the chunk size: "))
    input_list = [int(num) for num in input_list]
    chunk_iterator = ChunkIterator(input_list, chunk_size)
    print(f"\nChunks of Size {chunk_size}:")
```

```
    for chunk in chunk_iterator:
        print(chunk)
```

Q3: Create an iterator that iterates through a sequence in reverse order. Test it with a list of strings.

Program:

```
class ReverseIterator:
    def __init__(self, iterable):
        self.iterable = iterable
        self.index = len(iterable)

    def __iter__(self):
        return self

    def __next__(self):
        if self.index <= 0:
            raise StopIteration
        self.index -= 1
        return self.iterable[self.index]
if __name__ == "__main__":
    input_list = input("Enter a list of strings separated by spaces: ").split()
    reverse_iterator = ReverseIterator(input_list)
    print("\nSequence in Reverse Order:")
    for element in reverse_iterator:
        print(element)
```

…The End…