SET A:(Python)

```python
class Node:
    def __init__(self, vertex, weight, next_node=None):
        self.vertex = vertex
        self.weight = weight
        self.next = next_node


class Graph:
    def __init__(self, num_vertices):
        self.adj_list = [None] * num_vertices
        self.num_vertices = num_vertices

    def add(self, u, v, weight):

        node = Node(v, weight, self.adj_list[u])
        self.adj_list[u] = node

    def max_vertex_sum(self):
        max_weight = 0
        max_vertex = -1

        for i in range(self.num_vertices):
            current = self.adj_list[i]
            total_weight = 0
            while current!=None:
                total_weight += current.weight
                current = current.next

            if total_weight > max_weight:
                max_weight = total_weight
                max_vertex = i

        return max_vertex, max_weight


vertices = 6
graph = Graph(vertices)
graph.add(0, 1, 3)
graph.add(0, 2, 5)
graph.add(1, 3, 4)
graph.add(2, 3, 6)
graph.add(2, 4, 2)
graph.add(3, 4, 1)
graph.add(0, 4, 8)
graph.add(1, 2, 11)
graph.add(3, 5, 9)
vertex, sum = graph.max_vertex_sum()
print(vertex)
print(sum)
```

SET A:(Java)

```
class Node {
    int vertex;
    int weight;
    Node next;

    public Node(int vertex, int weight, Node next) {
        this.vertex = vertex;
        this.weight = weight;
        this.next = next;
    }
}

class Graph {
    private Node[] adjList;
    private int numVertices;

    public Graph(int numVertices) {
        this.numVertices = numVertices;
        this.adjList = new Node[numVertices];
    }

    public void add(int u, int v, int weight) {
        Node node = new Node(v, weight, adjList[u]);
        adjList[u] = node;
    }

    public int[] max_vertex_sum() {
        int maxWeight = 0;
        int maxVertex = -1;
        for (int i = 0; i < numVertices; i++) {
            Node current = adjList[i];
            int totalWeight = 0;

            while (current != null) {
                totalWeight += current.weight;
                current = current.next;
            }

            if (totalWeight > maxWeight) {
                maxWeight = totalWeight;
                maxVertex = i;
            }
        }

        return new int[]{maxVertex, maxWeight};
    }


}

public class Main{
      public static void main(String[] args) {
        int vertices = 6;
        Graph graph = new Graph(vertices);

        graph.add(0, 1, 3);
        graph.add(0, 2, 5);
        graph.add(1, 3, 4);
        graph.add(2, 3, 6);
        graph.add(2, 4, 2);
        graph.add(3, 4, 1);
        graph.add(0, 4, 8);
        graph.add(1, 2, 11);
        graph.add(3, 5, 9);

        int[] result = graph.max_vertex_sum();
        int vertex = result[0];
        int sum = result[1];

        System.out.println(vertex);
        System.out.println(sum);
    }
}
```

SET B(Python)

```python
class Node:
    def __init__(self, vertex, weight, next_node=None):
        self.vertex = vertex
        self.weight = weight
        self.next = next_node


class Graph:
    def __init__(self, num_vertices):
        self.adj_list = [None] * num_vertices
        self.num_vertices = num_vertices

    def add(self, u, v, weight):

        node = Node(v, weight, self.adj_list[u])
        self.adj_list[u] = node

    def max_vertex_product(self):
        max_product = 0
        max_vertex = -1

        for i in range(self.num_vertices):
            current = self.adj_list[i]
            product = 1


            while current!=None:
                product *= current.weight
                current = current.next

            if product > max_product:
                max_product = product
                max_vertex = i

        return max_vertex, max_product
```

```
num_vertices = 6
graph = Graph(num_vertices)
graph.add(0, 1, 3)
graph.add(0, 2, 5)
graph.add(1, 3, 4)
graph.add(2, 3, 6)
graph.add(2, 4, 2)
graph.add(3, 4, 1)
graph.add(0, 4, 8)
graph.add(1, 2, 11)
graph.add(3, 5, 9)
graph.add(4, 5, 10)

vertex, product = graph.max_vertex_product()

print(vertex)
print(product)
```

SET B(Java)

```
class Node {
    int vertex;
    int weight;
    Node next;

    public Node(int vertex, int weight, Node next) {
        this.vertex = vertex;
        this.weight = weight;
        this.next = next;
    }
}

class Graph {
    private Node[] adjList;
    private int numVertices;

    public Graph(int numVertices) {
        this.numVertices = numVertices;
        this.adjList = new Node[numVertices];
    }
    public void add(int u, int v, int weight) {
        Node node = new Node(v, weight, adjList[u]);
        adjList[u] = node;
    }

    public int[] max_vertex_product() {
        int maxProduct = 0;
        int maxVertex = -1;

        for (int i = 0; i < numVertices; i++) {
            Node current = adjList[i];
            int product = 1;

            while (current != null) {
                product *= current.weight;
                current = current.next;
            }

            if (product > maxProduct) {
                maxProduct = product;
                maxVertex = i;
            }
        }

        return new int[]{maxVertex, maxProduct};
    }

}

public class Main{
      public static void main(String[] args) {
        int vertices = 6;
        Graph graph = new Graph(vertices);

        graph.add(0, 1, 3);
        graph.add(0, 2, 5);
        graph.add(1, 3, 4);
        graph.add(2, 3, 6);
        graph.add(2, 4, 2);
        graph.add(3, 4, 1);
        graph.add(0, 4, 8);
        graph.add(1, 2, 11);
        graph.add(3, 5, 9);
        int[] result = graph.max_vertex_product();
        int vertex = result[0];
        int product = result[1];

        System.out.println(vertex);
        System.out.println(product);



    }
}
```

Rubric

Draw the graph in the script	2 marks
Using array + Linked list	4 marks
Traverse the linked list and array	2 marks
Calculate Sum(SET A) or Product(SET B)	4 marks
Return the max sum or product and the corresponding vertex	3 marks