```
def test_create_graph():
    """
    Tests that a graph can be created.
    """
    graph = create_graph()
    assert isinstance(graph, dict)


def test_weight_edges():
    """
    Tests that the edges of a graph can be weighted.
    """
    graph = create_graph()
    weight_edges(graph)
    for edge in graph.edges:
        assert isinstance(edge.weight, int)


def test_create_user_list():
    """
    Tests that a list of all the podcasts that the user has listened to can be created.
    """
    user_list = create_user_list()
    assert isinstance(user_list, list)
```

```
def test_find_most_similar_podcasts():
    """
    Tests that the most similar podcasts to a given podcast can be found.
    """
    graph = create_graph()
    weight_edges(graph)
    user_list = create_user_list()
    most_similar_podcasts = find_most_similar_podcasts(
        graph, user_list
    )
    assert isinstance(most_similar_podcasts, list)


def test_add_most_similar_podcasts():
    """
    Tests that the most similar podcasts to a given podcast can be added to the user's list.
    """
    graph = create_graph()
    weight_edges(graph)
    user_list = create_user_list()
    add_most_similar_podcasts(graph, user_list)
    assert len(user_list) > 0


def test_repeat_steps():
    """
```

```
Tests that steps 5-6 can be repeated until the user's list contains the desired number of podcasts.

"""

graph = create_graph()

weight_edges(graph)

user_list = create_user_list()

repeat_steps(graph, user_list)

assert len(user_list) == 10
```