Streams from multiple HLS (HTTP Live Streaming) manifest files with multiple resolutions in a 30-second window

There are several classes defined: Segment, Playlist, and HLSManifest. The Segment class has two properties: resolution and url. The Playlist class has a vector of Segment objects and a method addSegment to add segments to the vector. The HLSManifest class has a map of Playlist objects, where the key is the resolution and the value is the Playlist object. It also has a method addPlaylist to add playlists to the map and a method `

```
#include <iostream>
#include <vector>
#include <map>
#include <fstream>
#include <sstream>

struct Segment {
  int resolution;
  std::string url;
};

class Playlist {
  public:
    std::vector<Segment> segments;
    void addSegment(Segment segment) {
      segments.push_back(segment);
    }
};

class HLSManifest {
  private:
    std::map<int, Playlist> playlists;
  public:
    void addPlaylist(int resolution, Playlist playlist) {
      playlists[resolution] = playlist;
    }
    std::string generateManifest() {
      std::string manifest = "#EXTM3U\n";
      for (const auto& playlist : playlists) {
        manifest += "#EXT-X-STREAM-INF:BANDWIDTH=" + std::to_string(playlist.first) + "\n";
        manifest += playlist.second.segments[0].url + "\n";
      }
      return manifest;
    }
};

std::vector<Segment> parseManifestFile(const std::string& filename) {
  std::vector<Segment> segments;
  std::ifstream file(filename);
  std::string line;
  while (std::getline(file, line)) {
    if (line.find("#EXT-X-STREAM-INF") != std::string::npos) {
      int bandwidth;
      std::stringstream ss(line.substr(line.find(":") + 1));
      ss >> bandwidth;
      std::getline(file, line);
      segments.push_back({ bandwidth, line });
    }
  }
  return segments;
}

int main() {
  HLSManifest hlsManifest;
  std::vector<std::string> filenames = { "manifest1.m3u8", "manifest2.m3u8" };
  for (const auto& filename : filenames) {
    std::vector<Segment> segments = parseManifestFile(filename);
    for (const auto& segment : segments) {
      Playlist playlist;
      playlist.addSegment(segment);
      hlsManifest.addPlaylist(segment.resolution, playlist);
    }
  }
  std::cout << hlsManifest.generateManifest() << std::endl;
  return 0;
}
```