EX.NO:	**MOVIE RECOMMENDER SYSTEM USING BINARY TREE**
DATE: / /2025	

AIM

To Implement a Movie recommender system using binary tree in python.

SOURCE CODE:

```
import csv
class MovieRecommender:
    def __init__(self, csv_file):
        self.movies = []
        self.load_csv(csv_file
    def load_csv(self, file):
        with open(file, 'r', encoding='utf-8') as f:
            for row in csv.DictReader(f):
                self.movies.append({
                    'title': row['title'],
                    'genres': row['genres'].lower().split('|'),
                    'year': int(row.get('year', 0)),
                    'rating': float(row.get('rating', 0))
                })
    def ask(self, question):
        print(f"\n{question}")
        return input("Answer (yes/no): ").strip().lower() in ['yes', 'y']
    def filter(self, movies, genre=None, year_min=None, year_max=None,
rating_min=None):
        if genre:
            movies = [m for m in movies if genre.lower() in m['genres']]
        if year_min:
            movies = [m for m in movies if m['year'] >= year_min]
        if year_max:
            movies = [m for m in movies if m['year'] <= year_max]
        if rating_min:
            movies = [m for m in movies if m['rating'] >= rating_min]
        return sorted(movies, key=lambda x: x['rating'], reverse=True)

    def recommend(self):
        movies = self.movies.copy(
        if self.ask("Do you prefer realistic movies over fantasy/sci-fi?"):
            if self.ask("Do you want action movies?"):
                movies = self.filter(movies, genre='action')
            else:
                movies = self.filter(movies, genre='drama')
        else:
            if self.ask("Do you like science fiction?"):
                movies = self.filter(movies, genre='sci-fi')
            else:
                if self.ask("Do you want horror movies?"):
                    movies = self.filter(movies, genre='horror')
                else:
                    movies = self.filter(movies, genre='comedy'
        if self.ask("Do you prefer recent movies (2010+)?"):
            movies = self.filter(movies, year_min=2010)
        if self.ask("Do you want highly rated movies only (8.0+)?"):
            movies = self.filter(movies, rating_min=8.0)
    def display(self, movies):
        print("\n" + "="*60)
        print("RECOMMENDATIONS")
        print("="*60)
        if not movies:
            print("No movies found. Try different preferences.")
def main():
    print("="*60)
    print("MOVIE RECOMMENDER SYSTEM")
    print("="*60)
    csv_file = input("\nEnter CSV filename (e.g., movies.csv): ").strip()
    try:
        recommender = MovieRecommender(csv_file)
        while True:
            recommendations = recommender.recommend()
            recommender.display(recommendations)
           if input("\nGet another recommendation? (yes/no): ").strip().lower() not in ['yes', 'y']:
                print("\nThank you! 🎬")
                break
    except FileNotFoundError:
        print(f"✗ Error: '{csv_file}' not found!")
        print("\nCreate a CSV file with columns: title, genres, year, rating")
        print("Example:")
        print("title,genres,year,rating")
        print("Inception,Action|Sci-Fi|Thriller,2010,8.8")
        print("The Matrix,Action|Sci-Fi,1999,8.7")
if __name__ == "__main__":
    main()
```

OUTPUT:

```
============================================================
MOVIE RECOMMENDER SYSTEM
============================================================

Enter CSV filename (e.g., movies.csv): movie.csv
✓ Loaded 20 movies


Do you prefer realistic movies over fantasy/sci-fi?
Answer (yes/no): no

Do you like science fiction?
Answer (yes/no): yes

Do you prefer recent movies (2010+)?
Answer (yes/no): yes

Do you want highly rated movies only (8.0+)?
Answer (yes/no): yes

============================================================
RECOMMENDATIONS
============================================================
1. Inception (2010) - ★ 8.8
   Genres: action, sci-fi, thriller

2. Interstellar (2014) - ★ 8.7
   Genres: adventure, drama, sci-fi

============================================================
```

RESULT:

The program has been successfully executed.