|  |
| --- |
| **CodSoft** |
| Sharon Joseph |
| **Task 1: Tic Tac Toe** |
| **Program:** Used Python |
| import random  def initialize\_board():  return [' ' for \_ in range(9)]  def display\_board(board):  print(board[0] + '|' + board[1] + '|' + board[2])  print('-+-+-')  print(board[3] + '|' + board[4] + '|' + board[5])  print('-+-+-')  print(board[6] + '|' + board[7] + '|' + board[8])  def is\_board\_full(board):  return ' ' not in board  def check\_winner(board, player):  for win\_combination in WINNING\_COMBINATIONS:  if all(board[i] == player for i in win\_combination):  return True  return False  def minimax(board, depth, player):  if player == AI:  best\_score = -float('inf')  for move in range(9):  if board[move] == ' ':  board[move] = AI  score = minimax(board, depth + 1, HUMAN)  board[move] = ' '  best\_score = max(score, best\_score)  return best\_score  else:  best\_score = float('inf')  for move in range(9):  if board[move] == ' ':  board[move] = HUMAN  score = minimax(board, depth + 1, AI)  board[move] = ' '  best\_score = min(score, best\_score)  return best\_score  def find\_best\_move(board):  best\_move = -1  best\_score = -float('inf')  for move in range(9):  if board[move] == ' ':  board[move] = AI  score = minimax(board, 0, HUMAN)  board[move] = ' '  if score > best\_score:  best\_score = score  best\_move = move  return best\_move  def play\_game():  board = initialize\_board()  current\_player = random.choice([HUMAN, AI])  print("Tic-Tac-Toe - You are '{}' and AI is '{}'".format(HUMAN, AI))  display\_board(board)  while True:  if current\_player == HUMAN:  move = int(input("Enter your move (0-8): "))  if 0 <= move <= 8 and board[move] == ' ':  board[move] = HUMAN  else:  print("Invalid move. Try again.")  continue  else:  move = find\_best\_move(board)  board[move] = AI  print("AI's move:")  display\_board(board)  if check\_winner(board, current\_player):  print(f"{current\_player} wins!")  break  elif is\_board\_full(board):  print("It's a draw!")  break  current\_player = HUMAN if current\_player == AI else AI  HUMAN = 'X'  AI = 'O'  WINNING\_COMBINATIONS = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6)]  if \_\_name\_\_ == "\_\_main\_\_":  play\_game() |
|  |
|  |
|  |

|  |
| --- |
| **Task 2: CHATBOT WITH RULE-BASED RESPONSES** |
| **Program:** Used Python |
| import re  # Define some predefined rules and responses  rules = {  r'hi|hello|hey': 'Hello there! How can I help you today?',  r'how are you': 'I am just a computer program, but thanks for asking!',  r'what is your name': "I'm a chatbot, so I don't have a name, but you can call me YourAssist.",  r'bye|goodbye': 'Goodbye! Have a great day!',  r'help': 'I can assist you with general information. Just ask me a question!',  r'(.\*) your name (.\*)': "I'm just a chatbot, but you can call me ChatGPT.",  r'(.\*) (age|old) (.\*)': "I don't have an age. I'm just a computer program.",  r'(.\*) (created|made) (.\*)': "I was created by a developer.",  r'(.\*) (thank you|thanks) (.\*)': "You're welcome!",  }  # Function to match user input to predefined rules and provide responses  def chatbot\_response(user\_input):  user\_input = user\_input.lower()  for pattern, response in rules.items():  if re.match(pattern, user\_input):  return response  # If no rules match, provide a default response  return "I'm sorry, I don't understand. You can type 'help' for assistance."  # Main loop to interact with the chatbot  while True:  user\_input = input("You: ")  if user\_input.lower() == 'exit':  print("Chatbot: Goodbye!")  break  response = chatbot\_response(user\_input)  print("Chatbot:", response) |
|  |
|  |
|  |

|  |
| --- |
| **Task 3: RECOMMENDATION SYSTEM** |
| **Program:** Used C |
| #include <stdio.h>  #include <string.h>  #define MAX\_MOVIES 5  #define MAX\_GENRES 5  struct Movie {  char title[100];  char genres[MAX\_GENRES][100];  int genreCount;  };  void recommendMovies(struct Movie movies[], int movieCount, char userPreferences[][100], int preferenceCount) {  printf("Recommended Movies:\n");  int i;  for (i = 0; i < movieCount; i++) {  int score = 0;  int j;  int k;  for (j = 0; j < movies[i].genreCount; j++) {  for (k = 0; k < preferenceCount; k++) {  if (strcmp(movies[i].genres[j], userPreferences[k]) == 0) {  score++;  }  }  }  if (score > 0) {  printf("%s\n", movies[i].title);  }  }  }  int main() {  printf(" ..... Geners ..... \n\t Action\n\t Thriller\n\t Comedy\n\t Family\n\t Love\n\t Drama\n");  struct Movie movies[MAX\_MOVIES];  int movieCount = 0;  strcpy(movies[movieCount].title, "Bruce Lee");  strcpy(movies[movieCount].genres[0], "Action");  movies[movieCount].genreCount = 1;  movieCount++;  strcpy(movies[movieCount].title, "Jumanji");  strcpy(movies[movieCount].genres[0], "Thriller");  movies[movieCount].genreCount = 1;  movieCount++;  strcpy(movies[movieCount].title, "Mr. Bean");  strcpy(movies[movieCount].genres[0], "Comedy");  strcpy(movies[movieCount].genres[1], "Family");  movies[movieCount].genreCount = 2;  movieCount++;  strcpy(movies[movieCount].title, "Titanic");  strcpy(movies[movieCount].genres[0], "Love");  strcpy(movies[movieCount].genres[1], "Drama");  movies[movieCount].genreCount = 2;  movieCount++;  char userPreferences[MAX\_GENRES][100];  int preferenceCount = 0;  printf("Enter your preferences one by one. Enter 'done' when you're finished.\n");  while (1) {  printf("Enter a genre (or 'done'): ");  char input[100];  scanf("%s", input);  if (strcmp(input, "done") == 0) {  break;  }  switch (input[0]) {  case 'A':  case 'a':  strcpy(userPreferences[preferenceCount], "Action");  break;  case 'T':  case 't':  strcpy(userPreferences[preferenceCount], "Thriller");  break;  case 'C':  case 'c':  case 'F':  case 'f':  strcpy(userPreferences[preferenceCount], "Comedy");  break;  case 'L':  case 'l':  case 'D':  case 'd':  strcpy(userPreferences[preferenceCount], "Love");  break;  default:  printf("Invalid genre. Please choose from the available genres.\n");  continue;  }  preferenceCount++;  }  recommendMovies(movies, movieCount, userPreferences, preferenceCount);  return 0;  } |
|  |
|  |