**Question 1 (7 marks)**

|  |  |
| --- | --- |
| 1 | # Men’s shoe sizing guide  men\_sizes = {  6: 24.0, 6.5: 24.5, 7: 25.0, 7.5: 25.5, 8: 26.0, 8.5: 26.5, 9: 27.0,  9.5: 27.5, 10: 28.0, 10.5: 28.5, 11: 29.0, 11.5: 29.5  }  # Ladies’ shoe sizing guide  women\_sizes = {  4.5: 21.5, 5: 22.0, 5.5: 22.5, 6: 23.0, 6.5: 23.5, 7: 24.0, 7.5: 24.5,  8: 25.0, 8.5: 25.5, 9: 26.0  }  # Foot width recommendation guide  width\_recommendations = {  "Wide": 2.35,  "Standard": 2.65  }  gender = input("Enter gender (M/F): ").strip().lower()  foot\_length = float(input("Enter foot length (cm): "))  foot\_width = float(input("Enter foot width (cm): "))  if foot\_length < 21.1 or foot\_length > 29.5:  print("Call Customer Service for custom-made shoe")  else:  if gender == 'm':  if foot\_length < 23.6:  for size, length in women\_sizes.items():  if foot\_length <= length:  print(f"Recommended shoe size is Ladies US {size}")  # Comment for recommended size range  print(f"Foot is standard width, and ladies US {size} can accommodate up to {length}cm")  break  else:  for size, length in men\_sizes.items():  if foot\_length <= length:  if foot\_width < width\_recommendations["Wide"]:  print(f"Recommended shoe size is Men's US {size + 0.5}")  # Comment for recommended size range  print(f"Foot is standard width, and men’s US {size} can accommodate up to {length}cm")  print("Foot is wide. Hence, recommends next larger size")  elif foot\_width > width\_recommendations["Standard"]:  print(f"Recommended shoe size is Men's US {size - 0.5}")  # Comment for recommended size range  print(f"Foot is standard width, and men’s US {size} can accommodate up to {length}cm")  print("Foot is narrow. Hence, recommends next smaller size")  else:  print(f"Your shoe size is Men's US {size}")  break  elif gender == 'f':  if foot\_length > 26.0:  for size, length in men\_sizes.items():  if foot\_length <= length:  print(f"Recommended shoe size is Men's US {size}")  # Comment for recommended size range  print(f"Ladies US {size} can accommodate up to {length}cm, but foot is wide. Hence, recommends next larger size")  break  else:  for size, length in women\_sizes.items():  if foot\_length <= length:  if foot\_width < width\_recommendations["Wide"]:  print(f"Recommended shoe size is Ladies' US {size + 0.5}")  # Comment for recommended size range  print(f"Ladies US {size} can accommodate up to {length}cm, but foot is wide. Hence, recommends next larger size")  elif foot\_width > width\_recommendations["Standard"]:  if size == 4.5:  print("Call Customer Service for custom-made shoe")  # Comment for recommended size range  print(f"Ladies US {size} can accommodate up to {length}cm, but foot is narrow. Hence, recommends next smaller size, which is not available. Hence custom-made")  else:  print(f"Recommended shoe size is Ladies' US {size - 0.5}")  # Comment for recommended size range  print(f"Ladies US {size} can accommodate up to {length}cm, but foot is narrow. Hence, recommends next smaller size")  else:  print(f"Your shoe size is Ladies' US {size}")  break |

**Question 2 (18 marks)**

|  |  |
| --- | --- |
| 2a) | # Café menu items  menu = {  "A": {"item": "Soup of the day", "price": 4.50},  "B": {"item": "Garden Salad", "price": 5.50},  "C": {"item": "BLT Sandwich", "price": 6.50}  }  # Displaying the menu  print("<< Café Menu >>")  for key, value in menu.items():  print(f"{key}. {value['item']}")  print("X. Exit")  # Initializing variables  order = []  total\_price = 0  # Processing orders  while True:  user\_input = input("Enter your order: ")  if user\_input.upper() == 'X':  break  elif user\_input.upper() in menu.keys():  order.append(user\_input.upper())  else:  print("Invalid input. Please try again.")  # Calculating the total price  for item in order:  total\_price += menu[item]["price"]  # Displaying the total price  print(f"Thank you, please pay ${total\_price:.2f}") |
| 2b) | # Café menu items  menu = {  "A": {"item": "Soup of the day", "price": 4.50},  "B": {"item": "Garden Salad", "price": 5.50},  "C": {"item": "BLT Sandwich", "price": 6.50}  }  # Displaying the menu  print("<< Café Menu >>")  for key, value in menu.items():  print(f"{key}. {value['item']}")  print("X. Exit")  # Initializing variables  order = []  total\_price = 0  # Processing orders  while True:  user\_input = input("Enter your order: ")  if user\_input.upper() == 'X':  break  elif user\_input.upper() in menu.keys():  order.append(user\_input.upper())  else:  print("Invalid input. Please try again.")  # Calculating the total price  for item in order:  total\_price += menu[item]["price"]  # Displaying the total price  print(f"Total: ${total\_price:.2f}")  # Checking for discount eligibility  if total\_price > 20:  is\_member = input("Are you a member? (Y/N): ").upper()  if is\_member == 'Y':  total\_price \*= 0.9  # Displaying the final price to the customer  print(f"Thank you, please pay ${total\_price:.2f}") |
| 2c) | # Café menu items  menu = {  "A": {"item": "Soup of the day", "price": 4.50},  "B": {"item": "Garden Salad", "price": 5.50},  "C": {"item": "BLT Sandwich", "price": 6.50}  }  # Displaying the menu  print("<< Café Menu >>")  for key, value in menu.items():  print(f"{key}. {value['item']}")  print("X. Exit")  # Initializing variables  order = []  total\_price = 0  # Processing orders  while True:  user\_input = input("Enter your order: ")  if user\_input.upper() == 'X':  break  elif user\_input.upper() in menu.keys():  order.append(user\_input.upper())  else:  print("Invalid input. Please try again.")  # Calculate combo discounts  combo\_ab\_price = 9.50  combo\_abc\_price = 15.00  combo\_counts = {"A": 0, "B": 0, "C": 0}  for item in order:  combo\_counts[item] += 1  # Calculate total price with combo discounts  total\_price = combo\_counts["A"] \* menu["A"]["price"] + \  combo\_counts["B"] \* menu["B"]["price"] + \  combo\_counts["C"] \* menu["C"]["price"]  # Calculate the combo discount for A and B  combo\_ab\_count = min(combo\_counts["A"], combo\_counts["B"])  total\_price -= combo\_ab\_count \* (menu["A"]["price"] + menu["B"]["price"] - combo\_ab\_price)  # Calculate the combo discount for A, B, and C  combo\_abc\_count = min(combo\_counts["A"], combo\_counts["B"], combo\_counts["C"])  total\_price -= combo\_abc\_count \* (menu["A"]["price"] + menu["B"]["price"] + menu["C"]["price"] - combo\_abc\_price)  # Displaying the total price  print(f"Total: ${total\_price:.2f}")  # Checking for discount eligibility  if total\_price > 20:  is\_member = input("Are you a member? (Y/N): ").upper()  if is\_member == 'Y':  total\_price \*= 0.9  # Displaying the final price to the customer  print(f"Thank you, please pay ${total\_price:.2f}") |
|  | **Outputs of 2a:**        **Outputs of 2b:**        **Outputs of 2c:** |

**Question 3 (25 marks)**

|  |  |
| --- | --- |
| 3a) | import random  import time  players = []  hands = []  game\_over = False  def draw\_hand():  while True:  hand\_size = int(input("Starting hand (number) of digits: "))  if hand\_size % 2 == 0 and hand\_size > 3:  return [random.randint(0, 9) for \_ in range(hand\_size)]  else:  print("Starting hand must be even and more than 3.")  def check\_win(hand):  return len(hand) == 0  def evaluate\_expression(expression, result):  if expression[0] == '0' or expression[2] == '0':  return True  elif expression[1] == '+':  return int(expression[0]) + int(expression[2]) == result or int(expression[0]) + int(expression[2]) == 0  elif expression[1] == '-':  return int(expression[0]) - int(expression[2]) == result or int(expression[0]) - int(expression[2]) == 0  return False  def play\_round(player, result):  hand = hands[player]  print(f"{players[player]}'s hand: {hand}")  while True:  expression = input("Enter expression: ")  if not expression.strip():  print("Skipped")  break  if len(expression) != 3 or not (expression[0].isdigit() and expression[2].isdigit() and expression[1] in '+-'):  print("Expression is not in proper format.")  break  if int(expression[0]) not in hand or int(expression[2]) not in hand:  print("Digits not in hand.")  elif evaluate\_expression(expression, result):  hand.remove(int(expression[0]))  hand.remove(int(expression[2]))  print("Correct!")  break  else:  print("Incorrect!")  break  if check\_win(hand):  global game\_over  game\_over = True  print(f"No more digits in hand. {players[player]} wins this game!")  if \_\_name\_\_ == '\_\_main\_\_':  while len(players) < 2:  player = input("Enter player: ")  if not player:  print("Minimum 2 players.")  continue  if player.lower() in [p.lower() for p in players]:  print("No duplicate names... re-enter.")  continue  players.append(player)  hands.append(draw\_hand())  print("\nLet's play...\n")  round\_num = 1  while True:  round\_result = random.randint(0, 9)  print(f"Round {round\_num}: Result {round\_result}\n")  for idx, player in enumerate(players):  play\_round(idx, round\_result)  if game\_over:  break  if game\_over:  break  round\_num += 1 |
| 3b) | import random  import time  players = []  hands = []  game\_over = False  def draw\_hand():  while True:  hand\_size = int(input("Starting hand (number) of digits: "))  if hand\_size % 2 == 0 and hand\_size > 3:  return [random.randint(0, 9) for \_ in range(hand\_size)]  else:  print("Starting hand must be even and more than 3.")  def check\_win(hand):  return len(hand) == 0  def evaluate\_expression(expression, result):  if expression[0] == '0' or expression[2] == '0':  return True  elif expression[1] == '+':  return int(expression[0]) + int(expression[2]) == result or int(expression[0]) + int(expression[2]) == 0  elif expression[1] == '-':  return int(expression[0]) - int(expression[2]) == result or int(expression[0]) - int(expression[2]) == 0  return False  def play\_round(player, result, skip\_card):  hand = hands[player]  print(f"{players[player]}'s hand: {hand}")  while True:  expression = input("Enter expression: ")  if not expression.strip():  print("Skipped")  if not skip\_card:  for \_ in range(2):  hand.append(random.randint(0, 9))  return False #skip card not used  else:  return True #skip card used  if len(expression) != 3 or not (expression[0].isdigit() and expression[2].isdigit() and expression[1] in '+-'):  print("Expression is not in proper format.")  for \_ in range(2):  hand.append(random.randint(0, 9))  break  if int(expression[0]) not in hand or int(expression[2]) not in hand:  print("Digits not in hand.")  elif evaluate\_expression(expression, result):  hand.remove(int(expression[0]))  hand.remove(int(expression[2]))  print("Correct!")  break  else:  print("Incorrect!")  for \_ in range(2):  hand.append(random.randint(0, 9))  break  if check\_win(hand):  global game\_over  game\_over = True  print(f"No more digits in hand. {players[player]} wins this game!")  if \_\_name\_\_ == '\_\_main\_\_':  while len(players) < 2:  player = input("Enter player: ")  if not player:  print("Minimum 2 players.")  continue  if player.lower() in [p.lower() for p in players]:  print("No duplicate names... re-enter.")  continue  players.append(player)  hands.append(draw\_hand())  print("\nLet's play...\n")  games\_won = [0 for \_ in range(len(players))]  skip\_card = [False for \_ in range(len(players))]  while True:  print("Games won:\n " + "\n ".join([f"{players[i]}: {games\_won[i]}" for i in range(len(players))]) + "\n")  round\_num = 1  while True:  round\_result = random.randint(0, 9)  print(f"Round {round\_num}: Result {round\_result}\n")  for idx, player in enumerate(players):  skip\_used = play\_round(idx, round\_result, skip\_card[idx])  if skip\_used:  skip\_card[idx] = False  if game\_over:  games\_won[idx] += 1  skip\_card[1-idx] = True  skip\_card[idx] = False  break  if game\_over:  break  round\_num += 1  if games\_won[0] == 3:  print(f"{players[0]} is the overall winner!")  break  elif games\_won[1] == 3:  print(f"{players[1]} is the overall winner!")  break  else:  print("Starting new game...\n")  game\_over = False  hands = []  hands = [draw\_hand() for \_ in range(len(players))] |
|  | **Outputs of 3a:**        **Outputs of 3b:** |