**Code:**

1. class HashTable1:

2.     """linear Probing Without Replacement"""

3.

4.     def \_\_init\_\_(self, size: int) -> None:

5.         self.record = []

6.         self.m = size

7.

8.         # initialize all records with 0

9.         for \_ in range(size):

10.             self.record.append([0, ""])  # [tel,name]

11.

12.     def display\_table(self) -> None:

13.         print("Hash Table Using Linear Probing Without Replacement")

14.         for i in range(len(self.record)):

15.             print(i, self.record[i])

16.

17.     def hash\_function(self, tel: int) -> int:

18.         key = (tel % self.m)

19.         return key

20.

21.     def generate\_table(self, recs: list[list]) -> None:

22.         for rec in recs:

23.             self.insert\_rec(rec)

24.

25.     def insert\_rec(self, rec: list) -> None:

26.         key = self.hash\_function(rec[0])

27.         if (self.record[key][0] == 0):

28.             # no collision

29.             self.record[key][0] = rec[0]

30.             self.record[key][1] = rec[1]

31.         else:  # collision

32.             while (self.record[key][0] != 0):

33.                 key = ((key+1) % self.m)

34.

35.             self.record[key][0] = rec[0]

36.             self.record[key][1] = rec[1]

37.

38. class HashTable2:

39.     """linear Probing With Replacement"""

40.

41.     def \_\_init\_\_(self, size: int) -> None:

42.         self.record = []

43.         self.m = size

44.

45.         # initialize all records with 0 and -1 link

46.         for \_ in range(size):

47.             self.record.append([0, "", -1])  # [tel,name,link]

48.

49.     def display\_table(self) -> None:

50.         print("Hash Table Using Linear Probing With Replacement")

51.         for i in range(len(self.record)):

52.             print(i, self.record[i])

53.

54.     def hash\_function(self, tel: int) -> int:

55.         key = (tel % self.m)

56.         return key

57.

58.     def generate\_table(self, recs: list[list]) -> None:

59.         for rec in recs:

60.             self.insert\_rec(rec)

61.

62.     def insert\_rec(self, rec: list) -> None:

63.         key = self.hash\_function(rec[0])

64.         if (self.record[key][0] == 0):

65.             # no collision

66.             self.record[key][0] = rec[0]

67.             self.record[key][1] = rec[1]

68.             self.record[key][2] = -1

69.         else:  # collision

70.             if (self.hash\_function(self.record[key][0]) == key):

71.                 # create link

72.                 last\_elmt = key

73.                 while (self.record[last\_elmt][2] != -1):

74.                     last\_elmt = self.record[last\_elmt][2]

75.                 k = last\_elmt

76.                 while (self.record[k][0] != 0):

77.                     k = ((k+1) % self.m)

78.                 self.record[last\_elmt][2] = k

79.                 self.record[k][0] = rec[0]

80.                 self.record[k][1] = rec[1]

81.                 self.record[k][2] = -1

82.             else:  # replacement

83.                 # find last link

84.                 for i in range(self.m):

85.                     if (self.record[i][2] == key):

86.                         prev\_link\_key = i

87.

88.                 old\_rec\_tel = self.record[key][0]

89.                 old\_rec\_name = self.record[key][1]

90.                 old\_rec\_link = self.record[key][2]

91.

92.                 self.record[key][0] = rec[0]

93.                 self.record[key][1] = rec[1]

94.                 self.record[key][2] = -1

95.

96.                 k = key

97.                 while (self.record[k][0] != 0):

98.                     k = ((k+1) % self.m)

99.

100.                 self.record[prev\_link\_key][2] = k

101.                 self.record[k][0] = old\_rec\_tel

102.                 self.record[k][1] = old\_rec\_name

103.                 self.record[k][2] = old\_rec\_link

104.

105. class HashTable3:

106.     """Double hashing"""

107.

108.     def \_\_init\_\_(self, size: int) -> None:

109.         self.record = []

110.         self.m = size

111.

112.         # initialize all records with 0

113.         for \_ in range(size):

114.             self.record.append([0, ""])  # [tel,name]

115.

116.         if (size <= 3):

117.             self.prime = size

118.         else:

119.             prime = [2, 3]

120.             for i in range(size):

121.                 for j in prime:

122.                     if (i % j == 0):

123.                         p = False

124.                         break

125.                 if (p):

126.                     prime.append(i)

127.             self.prime = prime[-1]

128.

129.     def hash1(self, key: int) -> int:

130.         return (key % self.m)

131.

132.     def hash2(self, key: int) -> int:

133.         return (self.prime - (key % self.prime))

134.

135.     def display\_table(self) -> None:

136.         print("Hash Table Using Double Hashing")

137.         for i in range(len(self.record)):

138.             print(i, self.record[i])

139.

140.     def generate\_table(self, recs: list[list]) -> None:

141.         for rec in recs:

142.             self.insert\_rec(rec)

143.

144.     def insert\_rec(self, rec: list) -> None:

145.         i = 0

146.         key = self.hash1(rec[0])

147.         k2 = (key + i\*self.hash2(rec[0])) % self.m

148.         while (self.record[k2][0] != 0):

149.             k2 = (key + i\*self.hash2(rec[0])) % self.m

150.             i += 1

151.         self.record[k2][0] = rec[0]

152.         self.record[k2][1] = rec[1]

153.

154. def input\_records(n: int) -> list[list]:

155.     records = []

156.     for i in range(n):

157.         name = input("Enter Name of the person:")

158.         tel = int(input("Enter Telephone Number:"))

159.         records.append([tel, name])

160.     return records

161.

162. def main() -> None:

163.     n = int(input("Enter the total number of records:"))

164.     records = input\_records(n)

165.     ch = 1

166.     while(ch != 5):

167.         print("MENU")

168.         print("1. Input Records")

169.         print("2. Use linear Probing Without Replacement")

170.         print("3. Use linear Probing With Replacement")

171.         print("4. Use Double Hashing")

172.         print("5. Exit")

173.

174.         ch = int(input("Enter your choice:"))

175.         match (ch):

176.             case 1:

177.                 n = int(input("Enter the total number of records:"))

178.                 records = input\_records(n)

179.             case 2:

180.                 t1 = HashTable1(n)

181.                 t1.generate\_table(records)

182.                 t1.display\_table()

183.             case 3:

184.                 t2 = HashTable2(n)

185.                 t2.generate\_table(records)

186.                 t2.display\_table()

187.             case 4:

188.                 t3 = HashTable3(n)

189.                 t3.generate\_table(records)

190.                 t3.display\_table()

191.             case 5:

192.                 print("Thank you !")

193.             case default:

194.                 print("Invalid Choice")

195.

196. if \_\_name\_\_ == "\_\_main\_\_":

197.     main()

**Output:**

PS D:\Coding\College-code-sem4> python .\DSAL\dsal\_a1\_hashing.py

Enter the total number of records:5

Enter Name of the person:sahil

Enter Telephone Number:9874515963

Enter Name of the person:akhilesh

Enter Telephone Number:9764512783

Enter Name of the person:yash

Enter Telephone Number:7851226664

Enter Name of the person:omkar

Enter Telephone Number:8451225555

Enter Name of the person:kaustubh

Enter Telephone Number:9461564622

MENU

1. Input Records

2. Use linear Probing Without Replacement

3. Use linear Probing With Replacement

4. Use Double Hashing

5. Exit

Enter your choice:2

Hash Table Using Linear Probing Without Replacement

0 [7851226664, 'yash']

1 [8451225555, 'omkar']

2 [9461564622, 'kaustubh']

3 [9874515963, 'sahil']

4 [9764512783, 'akhilesh']

MENU

1. Input Records

2. Use linear Probing Without Replacement

3. Use linear Probing With Replacement

4. Use Double Hashing

5. Exit

Enter your choice:3

Hash Table Using Linear Probing With Replacement

0 [8451225555, 'omkar', -1]

1 [9764512783, 'akhilesh', -1]

2 [9461564622, 'kaustubh', -1]

3 [9874515963, 'sahil', 1]

4 [7851226664, 'yash', -1]

MENU

1. Input Records

2. Use linear Probing Without Replacement

3. Use linear Probing With Replacement

4. Use Double Hashing

5. Exit

Enter your choice:4

Hash Table Using Double Hashing

0 [9764512783, 'akhilesh']

1 [8451225555, 'omkar']

2 [9461564622, 'kaustubh']

3 [9874515963, 'sahil']

4 [7851226664, 'yash']

MENU

1. Input Records

2. Use linear Probing Without Replacement

3. Use linear Probing With Replacement

4. Use Double Hashing

5. Exit

Enter your choice:9

Invalid Choice

MENU

1. Input Records

2. Use linear Probing Without Replacement

3. Use linear Probing With Replacement

4. Use Double Hashing

5. Exit

Enter your choice:5

Thank you !

PS D:\Coding\College-code-sem4>