# **Essar International School, Surat**

## Computer Science

**(Practical File)**

**2020-21**

Submitted By : Akshat Aryan

Submitted To : Ms. Sadhana Guna

# **ESSAR INTERNATIONAL SCHOOL**

# **CERTIFICATE**

This is to certify that the practical work recorded by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_of class **XII Science** in **Computer Science** is an original work of the student and has been carried out regularly in the **Computer Laboratory** during the Academic Session **2020 – 2021**

Teacher’s Signature Principal’s Signature

Examiner’s Signature School Seal

Date : 20/10/2020

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Name | Pg no. | T. Sign |
| 1. | Armstrong Number Check | 1 |  |
| 2. | Prime Number Check | 2 |  |
| 3. | Perfect Number Check | 3 |  |
| 4. | Pascal Triangle Generator | 4 |  |
| 5. | Sorting of list | 5 |  |
| 6. | Binary Search | 6 |  |
| 7. | Inserting element in a list | 7 |  |
| 8. | Matrix Operations | 8 |  |
| 9. | Stack Operations | 13 |  |
| 10. | Binary Push | 15 |  |
| 11. | Stack: Minimum value | 16 |  |
| 12. | Postfix Expression Evaluation | 18 |  |
| 13. | Hospital: No. of visits | 20 |  |
| 14. | Read a file and count no. of lines starting with ‘w’ and ending with ‘e’ | 21 |  |
| 15. | Read a file and copy all lines starting with lower case to a new file | 22 |  |
| 16. | no of occurrences of "to" and "the" in a data file | 23 |  |
| 17. | copy contents of a source file and overwrite destination file | 24 |  |
| 18. | Data File Handling | 25 |  |
| 19. | CSV File Handling | 29 |  |
| 20. | Binary File Operations | 31 |  |
| 21. | Python-Mysql Connectivity | 33 |  |

36

Mysql Queries and Outputs

22.

Q1. Write a program to read a number and check if it is an Armstrong number by creating a function cntdig that receives the number and counts the number of digits and check function receives a number and checks if its Armstrong number.

def cntdig(x):

i = 0

while x > 0:

x = x // 10

i = i + 1

return (i)

def check(x):

dig = cntdig(x)

suma = 0

a = x

while a > 0:

r = a % 10

a = a // 10

suma = suma + r \*\* dig

if suma == x:

print("Armstrong")

else:

print("Not Armstrong")

a = int(input("Enter your number:"))

check(a)

**1**

Q2. Write a function that checks for all prime numbers between 2 and a number x.

def prime(x):

i = -1

for i in range(2, x):

if x % i == 0:

break

if i == (x - 1):

print(x, "is prime")

x = int(input("Enter your limit:"))

for j in range(2, x):

prime(j)

**2**

Q3. Write a function that receives a number and check if it is a perfect number.

def perfect(x):

suma = 0

for i in range(1, x-1):

if x % i == 0:

suma = suma + i

if suma == x:

print(x, "is perfect number")

def test(a, b):

for i in range(a, b+1):

perfect(i)

test(0, 100)

**3**

Q4. Write a function that receives a value n and then generates first n lines of Pascal’s Triangle in a list.

def pascal(n):

n = n

l1 = []

for i in range(0, n):

coff = 1

l2 = []

for k in range(0, i+1):

l2.append(coff)

coff = int((coff \* i - k) / (k + 1))

l1.appendl(2)

return(l1)

a = int(input("Enter number of rows:"))

print(pascal(a))

**4**

Q5. Write functions that sort the list of numbers in ascending and descending order each.

def sorta(l1):

for i in range(0, lenl1)):

small = l1[i]

pos = i

for j in range(i+1, lenl1)):

if l1[j] < small:

small = l1[j]

pos = j

x = l1[i]

l1[i] = l1[pos]

l1[pos] = x

return (l1)

def sortd(l1):

for k in range(0, len(l1)):

large = l1[k]

pos = k

for j in range(k+1, len(l1)):

if l1[j] > large:

large = l1[j]

pos = j

x = l1[k]

l1[k] = l1[pos]

l1[pos] = x

return l1)

l1 = [5,2,12,16,1,14]

print(sorta(l1))

print(sortd(l1))

**5**

Q6. Write a function that performs binary search of a number in a list.

def searcha(l1, x): #ascending

beg = 0

last = len(l1) - 1

while beg <= last:

mid = (beg + last) // 2

if x == l1[mid]:

return (mid)

elif x < l1[mid]:

last = mid - 1

elif x > l1[mid]:

beg = mid + 1

return ("Not Found")

def searchd(l1, x): #descending

beg = 0

last = (lenl1) - 1

while beg <= last:

mid = (beg + last) // 2

if x == l1[mid]:

return (mid)

elif x > l1[mid]:

last = mid - 1

elif x < l1[mid]:

beg = mid + 1

return ("Not Found")

l1 = [1, 2, 3, 4, 5]

l2 = [5, 4, 3, 2, 1]

a = int(input("Enter the number you want to search:"))

print("The postion of", a, "in list 1 is:", searcha(l1, a))

print("The postion of", a, "in list 2 is:", searchd(l2, a))

**6**

Q7. Write a function that receives a sorted list and inserts element in ascending/descending order.

def findpos(l1, x):

siz = len(l1)

if x < l1[0]:

return (0)

else:

pos = -1

for i in range(0, siz-1):

if l1[i] <= x and x <= l1[i+1]:

pos = i + 1

break

if pos == -1:

pos = siz

return (pos)

def shift(l1, pos):

l1.append(0)

siz = len(l1)

i = siz - 1

while i >=pos:

l1[i] = l1[i-1]

i = i - 1

lst1 = [1, 16, 23, 54, 69, 80]

a = int(input("Enter the number which you want to add to your list:"))

pos = findpos(lst1, a)

shift(lst1, pos)

lst1[pos] = a

print(lst1)

print("Position of insertion:", pos)

**7**

Q8. Write a menu driven program to add, subtract, multiply, transpose, find sum of both diagonal's elements, print the upper and lower triangle.

Print("For the Matrices->")

deg = int(input("Enter the degree of square matrix:"))

print("For first Matrix->")

m1 = []

for i in range(0, deg):

r = []

for j in range(0, deg):

x = int(input("Enter your element at ("+str(i)+","+str(j)+"):"))

r.append(x)

m1.append(r)

print("For second Matrix->")

m2 = []

for i in range(0, deg):

r = []

for j in range(0, deg):

x = int(input("Enter your element at ("+str(i)+","+str(j)+"):"))

r.append(x)

m2.append(r)

print("Your matrices are->")

print("Matrix 1=")

for i in range(0, deg):

for j in range(0, deg):

print(m1[i][j], end = " ")

print("")

print("Matrix 2=")

for i in range(0, deg):

for j in range(0, deg):

print(m2[i][j], end = " ")

print("")

def sum(m1, m2):

deg = len(m1)

m3 = [] #m3 = m1 + m2

**8**

for i in range(0, deg):

row = []

for j in range(0, deg):

x = m1[i][j] + m2[i][j]

row.append(x)

m3.append(row)

return (m3)

def diff(m1, m2):

deg = len(m1)

m3 = [] #m3 = m1 + m2

for i in range(0, deg):

row = []

for j in range(0, deg):

x = m1[i][j] - m2[i][j]

row.append(x)

m3.append(row)

return (m3)

def trans(m1):

m3 = []

for i in range(0, len(m1)):

row = []

for j in range(0, len(m1)):

x = m1[j][i]

row.append(x)

m3.append(row)

print("The transpose is:")

for i in range(0, len(m3)):

for j in range(0, len(m3)):

print(m3[i][j], end = " ")

print("")

def sumdiag(m1):

x = 0

for i in range(0, len(m1)):

x = x + m1[i][i]

j = len(m1)

y = 0

for i in range(0, len(m1)):

y = y + m1[i][j-1]

j = j - 1

return(x, y)

**9**

def tri(m1):

print("Lower triangle:")

for i in range(0, len(m1)):

for j in range(0, i+1):

print(m1[i][j], end = " ")

print ("")

print("Upper triangle:")

for i in range(0, len(m1)):

if i == 0:

for j in range(0, len(m1)):

print(m1[i][j], end = " ")

print ("")

else:

for j in range(0, lenm1)):

if i > j:

print(" ", end = " ")

else:

print(m1[i][j], end = " ")

print ("")

def prod(m1, m2):

m3 = []

for i in range(0, len(m1)):

row = []

for j in range(0, len(m1)):

x = 0

for k in range(0, len(m2)):

x = x + m1[i][k] \* m2[k][j]

row.append(x)

m3.append(row)

return(m3)

print("Your choices are: \n1. add(+) \n2. subtract(-) \n3. multiply(\*) \n4. transpose(trans) \n5. Print upper and lower triangle(tri) \n6. Sum of Diagonals(sumd) \n7. Exit(exit)")

n = input("Enter your choice:")

while n != "exit" and n != "7":

if n == "+" or n == "1":

x = sum(m1, m2)

print("Sum =")

**10**

for i in range(0, len(x)):

for j in range(0, len(x)):

print(x[i][j], end = " ")

print("")

elif n == "-" or n == "2":

x = diff(m1, m2)

print("Difference =")

for i in range(0, len(x)):

for j in range(0, len(x)):

print(x[i][j], end = " ")

print("")

elif n == "\*" or n == "3":

x = prod(m1, m2)

print("Product =")

for i in range(0, len(x)):

for j in range(0, len(x)):

print(x[i][j], end = " ")

print("")

elif n == "trans" or n == "4":

condition = 0

while condition == 0:

x = int(input("Which matrix do you want to transpose (1 or 2):"))

if x == 1:

trans(m1)

condition = 1

elif x == 2:

trans(m2)

condition = 1

else:

print("Please input 1 or 2")

elif n == "tri" or n == "5":

condition = 0

while condition == 0:

x = int(input("Which matrix do you want to print triangles of (1 or 2):"))

if x == 1:

tri(m1)

condition = 1

elif x == 2:

**11**

tri(m2)

condition = 1

else:

print("Please input 1 or 2")

elif n == "sumd" or n == "6":

condition = 0

while condition == 0:

x = int(input("Which matrix do you want the sum of diagonals of(1 or 2):"))

if x == 1:

y = sumdiag(m1)

(a, b) = y

print("Sum of Left Diagonal =", a)

print("Sum of Right Diagonal =", b)

condition = 1

elif x == 2:

y = sumdiag(m2)

(a, b) = y

print("Sum of Left Diagonal =", a)

print("Sum of Right Diagonal =", b)

condition = 1

else:

print("Please input 1 or 2")

print ("")

print("Your choices are: \n1. add(+) \n2. subtract(-) \n3. multiply(\*) \n4. transpose(trans) \n5. Print upper and lower triangle(tri) \n6. Sum of Diagonals(sumd) \n7. Exit(exit)")

n = input("Enter your choice:")

**12**

Q9. Write a Menu driven program to perform Stack operations.

def isempty(stk):

if stk == []:

return True

else:

return False

def push(stk, x):

stk.append(x)

top = len(stk) - 1

def spop(stk):

if isempty(stk):

print ("Underflow")

else:

item = stk.pop()

if len(stk) == 0:

top = None

else:

top = len(stk) - 1

return (item)

def peek(stk):

if isempty(stk):

return ("Underflow")

else:

top = len(stk) - 1

return stk[top]

def disp(stk):

if isempty(stk):

print("Empty")

else:

top = len(stk) - 1

for i in range(top, -1, -1):

print(stk[i])

stack = []

while True:

**13**

print("Your choices are: \n1. Push \n2. Pop \n3. Peek \n4. Display \n5. Exit")

ch = int(input("Enter your choice(1/2/3/4/5):"))

if ch == 1:

a = int(input("Enter element to push:"))

push(stack, a)

elif ch == 2:

itm = spop(stack)

elif ch == 3:

itm = peek(stack)

print ("Top =", itm)

elif ch == 4:

disp(stack)

elif ch == 5:

break

**14**

Q10. Write a program to read a number, convert it into a stack and Display the stack.

def isempty(stk):

if stk == []:

return True

else:

return False

def push(stk, x):

stk.append(x)

top = len(stk)

def disp(stk):

if isempty(stk):

print("Empty")

else:

top = len(stk) - 1

for i in range(top, -1, -1):

print(stk[i])

def binary(n):

a = n

x = 0

sum = 0

while a > 0:

r = a % 2

a = a // 2

sum = sum + r \* (10 \*\* x)

x = x + 1

return (sum)

s1 = []

a = int(input("Enter the number of inputs:"))

for i in range(0, a):

x = int(input("Enter your number:"))

y = binary(x)

push(s1, y)

print("Your stack is:")

disp(s1)

**15**

Q11. Write a program to perform push and pop operations on a stack and returns the smallest element in a stack.

def isempty(stk):

if stk == []:

return True

else:

return False

def push(stk, x):

stk.append(x)

top = len(stk) - 1

def spop(stk):

if isempty(stk):

print ("Underflow")

else:

item = stk.pop()

if len(stk) == 0:

top = None

else:

top = len(stk) - 1

return (item)

def peek(stk):

if isempty(stk):

return ("Underflow")

else:

top = len(stk) - 1

return stk[top]

def disp(stk):

if isempty(stk):

print("Empty")

else:

top = len(stk) - 1

for i in range(top, -1, -1):

print(stk[i])

def findmin(x, s2):

y = peek(s2)

if x == y:

itm2 = spop(s2)

**16**

s1 = []

s2 = []

while True:

print("Your choices are: \n1. Push \n2. Pop \n3. Peek \n4. Display \n5. Exit")

ch = int(input("Enter your choice(1/2/3/4/5):"))

if ch == 1:

a = int(input("Enter element to push:"))

if isempty(s1) == True:

push(s1, a)

push(s2, a)

else:

push(s1, a)

x = peek(s2)

if a < x:

push(s2, a)

elif ch == 2:

itm1 = spop(s1)

findmin(itm1, s2)

elif ch == 3:

itm = peek(s1)

print ("Top =", itm)

elif ch == 4:

disp(s1)

print("Minimum value =", peek(s2))

elif ch == 5:

break

**17**

Q12. Write a Program to evaluate Postfix expression to Infix expression using stacks.

def isempty(stk):

if stk == []:

return True

else:

return False

def push(stk, x):

stk.append(x)

top = len(stk) - 1

def spop(stk):

if isempty(stk):

print ("Underflow")

else:

item = stk.pop()

if len(stk) == 0:

top = None

else:

top = len(stk) - 1

return (item)

def peek(stk):

if isempty(stk):

return ("Underflow")

else:

top = len(stk) - 1

return top[stk]

l1 = ["2", "3", "+", "4", "5", "+", "\*", "5", "/"]

s1 = []

top1 = None

for i in range(0, len(l1)):

if l1[i].isdigit():

push(s1, int(l1[i]))

else:

op1 = spop(s1)

op2 = spop(s1)

**18**

if l1[i] == "+":

x = op2 + op1

elif l1[i] == "-":

x = op2 - op1

elif l1[i] == "\*":

x = op2 \* op1

elif l1[i] == "/":

x = op2 / op1

elif l1[i] == "^":

x = op2 \*\* op1

push(s1, x)

print (s1)

print("Answer =", spop(s1))

**19**

Q13. Write a program that finds the patient with most number of visits to a clinic from a list that represents the date of visits of each patient.

l1 = [[2, 6], [3, 10], [15], [23], [1, 8, 15, 19, 22], [14]]

#elements inside the nested lists contain the dates of visits of different patients. Hence length of each nested list is no. of visits.

def most(l1):

large = [l1[0]]

for i in range(1, len(l1)):

if len(l1[i]) > len(large[0]):

large = []

large.append(l1[i])

elif len(l1[i]) == len(large[0]):

large.append(l1[i])

return(large)

print("The patient that visited the most number of times visited on these days:", most(l1))

**20**

Q14. Write a program to read a file and count the number of lines that start with ‘w’ or end with ‘e’.

f1 = open(“newfile.txt”, 'r')

str = ' '

ctw = 0

cte = 1

while str:

str = f1.readline()

if str == '':

break

if str[0] == 'w':

ctw = ctw + 1

print(str)

elif str[-1] == 'e':

cte = cte + 1

print(str)

f1.close()

print("Lines starting with 'w':", ctw)

print("Lines ending with 'e':", cte)

**21**

Q15. Write a function that reads a file and copies all lines that start with a lowercase letter onto another file.

def copy(file):

f1 = open(file, 'r')

f2 = open("file2.txt", 'w')

x = ' '

while x:

x = f1.readline()

if x == '':

break

if x[0].islower():

f2.write(x)

f1.close()

f2.close()

file = "file1.txt"

copy(file)

f = open("file2.txt", 'r')

str = f.read()

print(str)

**22**

Q16. Write a program to count the no of occurrences of "to" and "the" in a data file.

f1 = open(“file1.txt”, 'r')

suma = 0

sumb = 0

x = ' '

while x:

x = f1.readline()

if x == "":

break

l = x.split(" ")

for i in l:

if i == "to":

suma = suma + 1

if i == "the":

sumb = sumb + 1

print("No. of 'to':", suma)

print("No. of 'the':", sumb)

**23**

Q17. Write a program to copy contents of a source file to a destination file. If any of the files don’t exist, abandon operation.

import os

if os.path.exists("source.txt"):

f1 = open("source.txt"):

a = input("Do you want to overwrite the file?(yes/no)")

if a == "yes":

f2 = open("destination.txt", "w")

str = f1.read()

f2.write(str)

print("file overwrite successful")

f2.close()

else:

print("You chose not to overwite. the process will be abandoned.")

f1.close()

else:

print("Destination file not found. the process will be abandoned.")

f1.close()

else:

print("Source file not found. the process will be abandoned.")

**24**

Q18. Write a menu driven program to perform Data File Handling.

import os

def add():

global txt

txt = ""

rno = int(input("Enter the Roll. No.:"))

nm = input("Enter the name:")

pct = int(input("Enter the Percentage:"))

txt = str(rno) + "," + nm + "," + str(pct) + "\n"

def prt():

size = os.path.getsize(file)

if size == 0:

print("File Empty")

else:

f1.seek(0)

str = " "

while str:

str = f1.readline()

if str == "":

break

print(str)

def search(x):

size = os.path.getsize(file)

if size == 0:

print("File Empty")

else:

f1.seek(0)

str = " "

while str:

str = f1.readline()

if str == "":

print("Roll No. not found")

break

l = str.split(",")

if int(l[0]) == x:

print(str)

**25**

break

def delete(x):

size = os.path.getsize(file)

if size == 0:

print("File Empty")

else:

global f1

f1.seek(0)

str1 = " "

l1 = []

while str1:

str1 = f1.readline()

if str1 == "":

break

l = str1.split(",")

l1.append(l)

curlen = len(l1)

for i in range(0, len(l1)):

item = ""

if int(l1[i][0]) == x:

item = str(l1[i][0]) + "," + l1[i][1] + "," + str(l1[i][2])

l1.pop(i)

break

newlen = len(l1)

if curlen == newlen:

print("Roll No. not found")

else:

f1.close()

f1 = open(file, "w")

record = ""

for i in range(0, len(l1)):

record = str(l1[i][0]) + "," + l1[i][1] + "," + str(l1[i][2])

f1.write(record)

print("Values:", item, "were deleted successfully")

f1.close()

f1 = open(file, "a+")

def modify(x, a, b):

size = os.path.getsize(file)

**26**

if size == 0:

print("File Empty")

else:

global f1

f1.seek(0)

str1 = " "

l1 = []

condition = 1

while str1:

str1 = f1.readline()

if str1 == "":

break

l = str1.split(",")

l1.append(l)

for i in range(0, len(l1)):

item = ""

if int(l1[i][0]) == x:

condition = 0

item1 = str(l1[i][0]) + "," + l1[i][1] + "," + str(l1[i][2])

l1[i][1] = a

l1[i][2] = str(b) + "\n"

item2 = str(l1[i][0]) + "," + l1[i][1] + "," + str(l1[i][2])

if condition == 1:

print("Roll. no not found")

else:

f1.close()

f1 = open(file, "w")

record = ""

for i in range(0,len(l1)):

record = str(l1[i][0]) + "," + l1[i][1] + "," + str(l1[i][2])

f1.write(record)

print("Values:", item1, "were modified successfully into", item2)

f1.close()

f1 = open(file, "a+")

file = "dfh.txt"

txt = ""

f1 = open(file, "a+")

**27**

while True:

print("Your choices are: \n1. Add \n2. Edit \n3. Search \n4. Delete \n5. Print \n6. Exit")

x = int(input("Enter option number:"))

if x == 1:

while True:

add()

print("Data to be added is:", txt)

f1.write(txt)

print("Data added successfully")

ch = input("Do you want to add more data?(yes/no)")

if ch == "no":

break

f1.close()

f1 = open(file, "a+")

elif x == 2:

a = int(input("Enter the Roll No. you have to modify:"))

b = input("Enter new name:")

c = input("Enter new percentage:")

modify(a, b, c)

elif x == 3:

a = int(input("Enter the Roll No. you have to search for:"))

search(a)

elif x == 4:

a = int(input("Enter the Roll No. you have to delete:"))

delete(a)

elif x == 5:

prt()

elif x == 6:

break

f1.close()

**28**

Q19. Write a menu driven program to perform csv file handling.

import csv

stu = []

fields = ["roll no", "name", "percentage"]

file = "file1.csv"

def add():

while True:

rec = []

rec.append(int(input("Enter Roll. no.:")))

rec.append(input("Enter name:"))

rec.append(int(input("Enter percentage:")))

stu.append(rec)

ch = input("Continue adding more records?(y/n):")

if ch == "n":

break

while True:

print("Your choices are: \n1. Add \n2. Search \n3. Print \n4. Delete \n5. Exit")

x = int(input("Enter option number:"))

if x == 1:

add()

with open(file, 'a', newline = "") as adddata:

w = csv.writer(adddata)

w.writerow(fields)

w.writerows(stu)

if x == 2:

x = input("Enter the roll no. to search:")

c = 0

with open(file, 'r') as searchdata:

r = csv.reader(searchdata)

for i in r:

if i[0] == x:

c = 1

print("Found:", i)

if c == 0:

print("Not Found")

if x == 3:

**29**

with open(file, 'r') as printdata:

r = csv.reader(printdata)

for i in r:

print (i)

if x == 4:

x = input("Enter the roll no. to delete:")

c = 0

newlist = []

with open(file, 'r') as delete:

r = csv.reader(delete)

for i in r:

if i[0] == x:

c = 1

if i[0] != x:

newlist.append(i)

if c == 0:

print("Record not found")

else:

with open(file, 'w', newline = "") as deladd:

w = csv.writer(deladd)

w.writerow(fields)

w.writerows(newlist)

if x == 5:

break

**30**

Q20. Write a menu driven program to perform Binary File Operation.

import pickle

list = []

while True:

roll = input("Enter Roll. no.:")

sname = input("Enter name:")

student = {"roll":roll, "name":sname}

list.append(student)

ch = input("Continue adding more records?(y/n):")

if ch == "n":

break

f1 = "student.dat"

file = open(f1, "wb")

pickle.dump(list,file)

file.close()

#read binary file

file = open(f1, "rb")

list = pickle.load(file)

print(list)

file.close()

#search

name = input("Enter the name you want to search for:")

file = open(f1, "rb")

list = pickle.load(file)

file.close()

found = 0

for x in list:

if name in x["name"]:

found = 1

print ("Found in binary file" if found == 1 else "Not found")

#update

name = input("Enter the name you want to update:")

file = open(f1, "rb+")

**31**

list = pickle.load(file)

found = 0

for x in list:

if name in x["name"]:

found = 1

x["name"] = input("Enter new name:")

if found == 1:

file.seek(0)

pickle.dump(list, file)

print("Record updated")

else:

print("Name doesnt exist")

file.close()

file = open(f1, "rb")

list = pickle.load(file)

print(list)

file.close()

#delete

name = input("Enter the name you want to delete:")

file = open(f1, "rb+")

list = pickle.load(file)

found = 0

lst = []

for x in list:

if name not in x["name"]:

lst.append(x)

else:

found = 1

if found == 1:

file.seek(0)

pickle.dump(lst, file)

print("Record Deleted")

else:

print("Name not found")

file.close()

file = open(f1, "rb")

list = pickle.load(file)

print(list)

file.close()

**32**

Q21. Write a program on Python-Mysql Connectivity.

import mysql.connector as sqltor

mycon = sqltor.connect(host = "localhost", user = "root", passwd = "mysql", database = "test")

if mycon.is\_connected():

print("Connected Successfully")

c1=mycon.cursor()

n = -1

while True:

print("Your choices are: \n1. Add \n2. Modify \n3. Delete \n4. Ask Queries \n5. Exit")

n = input("Enter your choice:")

if n == '1':

while True:

i1 = int(input("Enter Tno.:"))

i2 = input("Enter Tname:")

i3 = int(input("Enter Salary:"))

i4 = input("Enter Area:")

i5 = int(input("Enter Age:"))

i6 = input("Enter the Grade:")

i7 = input("Enter the department:")

st = "insert into test values({}, '{}', {}, '{}', {}, '{}', '{}')".format(i1, i2, i3, i4, i5, i6, i7)

c1.execute(st)

ch = input("input 'y' to enter another row, 'n' to finish addition of data:")

if ch == 'n':

break

mycon.commit()

if n == '2':

a = int(input("Enter the Tno. of the row you need to modify:"))

b = int(input("Enter the % increase in the salary:"))

**33**

st = "update test set salary = salary + (({} / 100) \* salary) where tno = {};".format(b, a)

c1.execute(st)

mycon.commit()

if n == '3':

a = int(input("Enter the Tno. of the row you need to delete:"))

st = "delete from test where tno = {};".format(a)

c1.execute(st)

mycon.commit()

if n == "4":

print("Please select one of the following Querie: \n1. Display details for particular dept \n2. Display all details for age in a range \n3. Display all details for a salary")

a = int(input("Enter option number:"))

if a == 1:

dept = input("Enter the Department name:")

st = "select \* from test where dept = '{}'".format(dept)

c1.execute(st)

data = c1.fetchall()

print("Your data is:")

for row in data:

print(row)

if a == 2:

low = int(input("Enter the lower age limit:"))

up = int(input("Enter the upper age limit:"))

st = "select \* from test where age between {} and {}".format(low, up)

c1.execute(st)

data = c1.fetchall()

print("Your data is:")

for row in data:

print(row)

**34**

if a == 3:

sal = int(input("Enter the salary:"))

st = "select \* from test where salary = {}".format(sal)

c1.execute(st)

data = c1.fetchall()

print("Your data is:")

for row in data:

print(row)

if n == 5:

break

mycon.close()

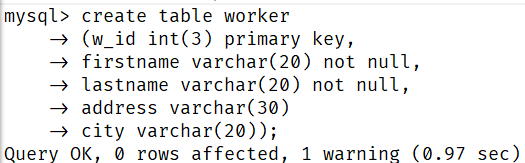
**35**

Q22. MySQL: create tables and write the output of the following queries.

1) Create table ‘worker’.

a. w\_id is primary key.

b. firstnname, lastname are not null.

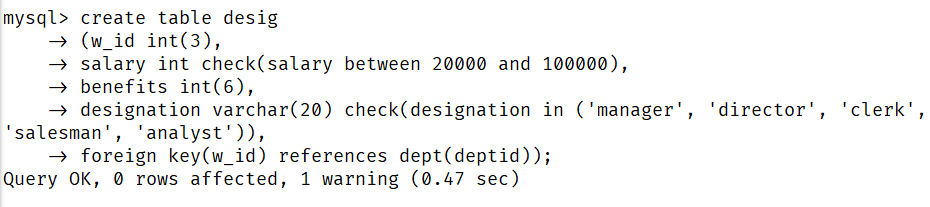


2) Create table ‘desig’.

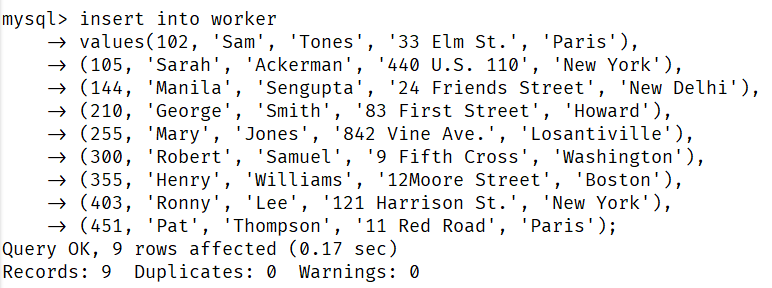
a. w\_id is foreign key.

b. salary is of range 20000 to 100000

c. designation should be – Manager, Director, Clerk, Salesman, Analyst

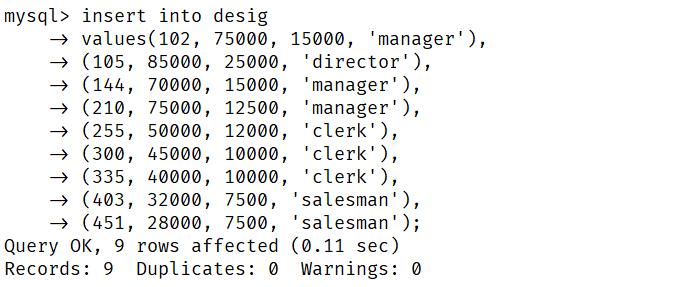


3) Add data to ‘worker’.

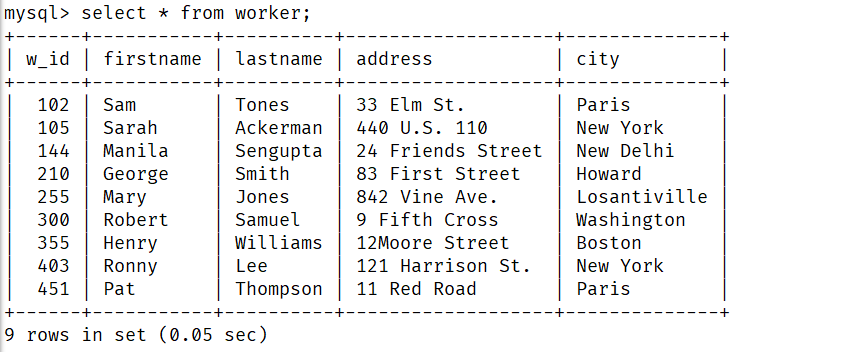


**36**

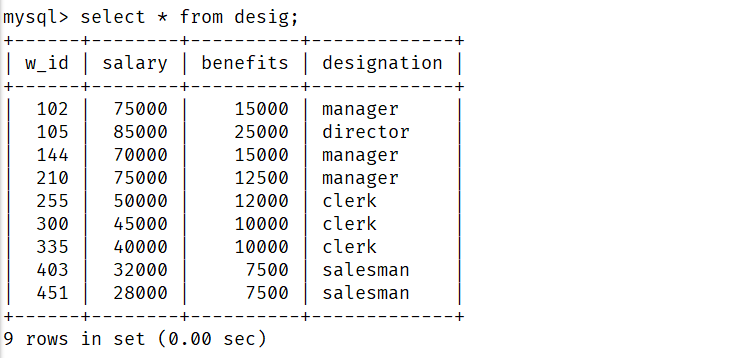
4) Add data to ‘desig’.



5) Display worker table.

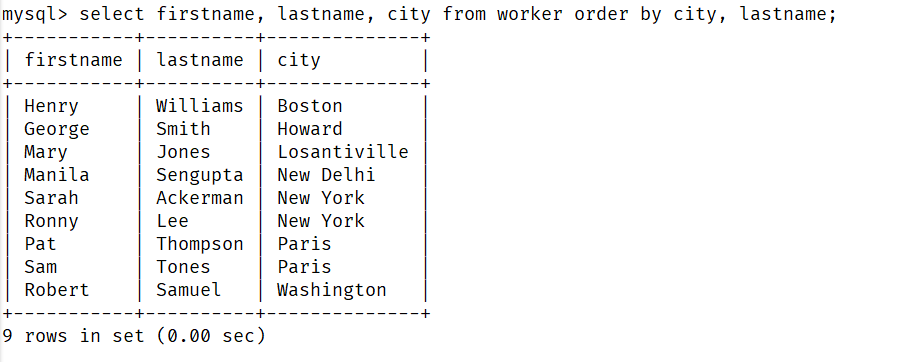


6) Display desig table.

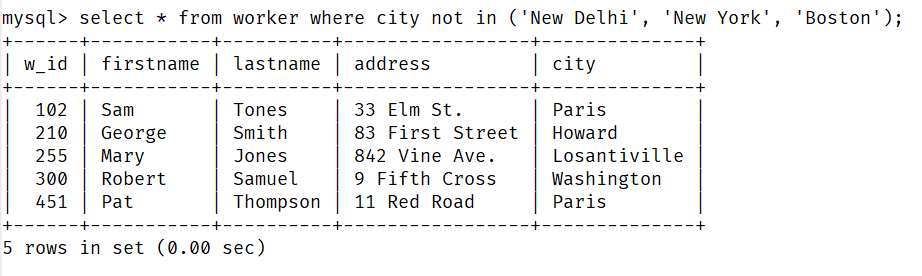


**37**

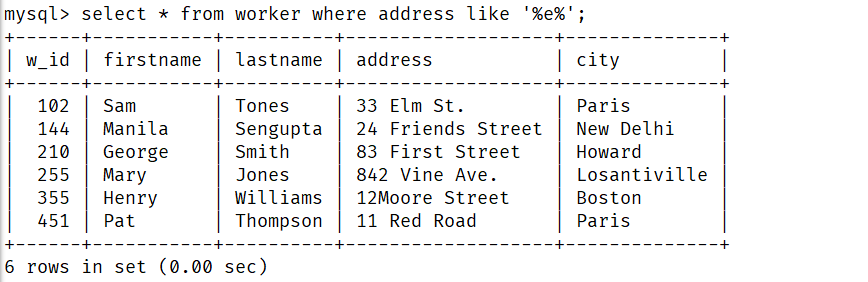
7) Display Firstname sorted on Lastname within city.



8) Display all workers who are not in ‘New Delhi’, ‘New York’ and ‘Boston’.

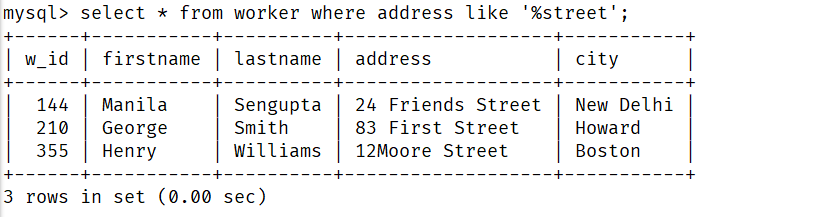


9) Display all workers whose address contains the letter ‘e’.

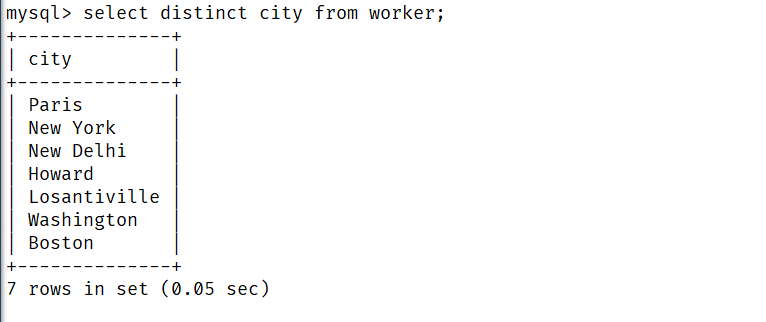


**38**

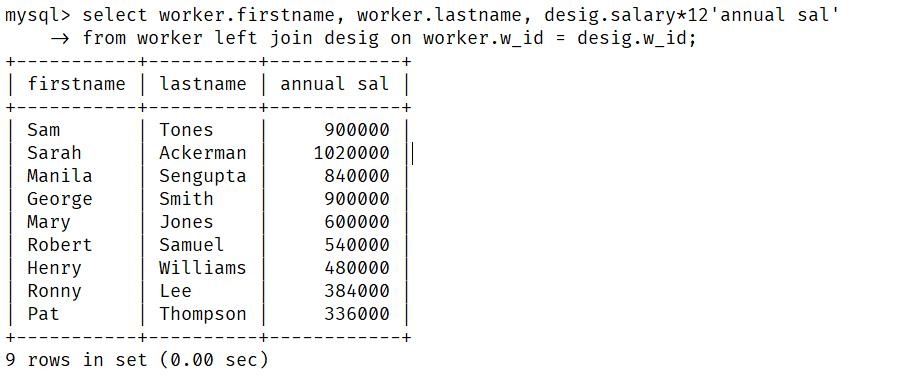
10) Display all workers whose address ends with ‘street’.



11) Display unique city names for workers.

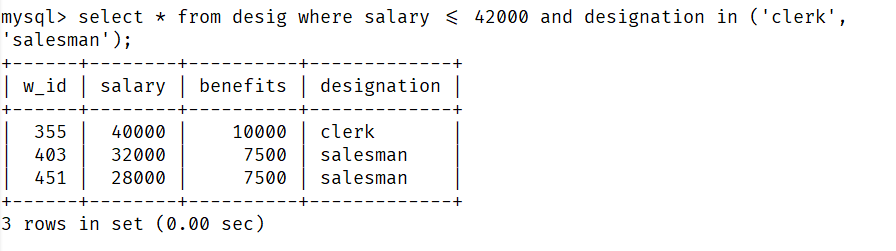


12) Display worker name, annual salary for all workers, give proper heading to annual salary.

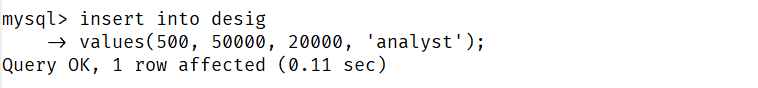


**39**

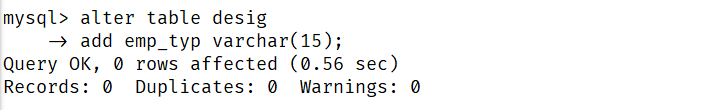
13) Display details of all Clerks and Salesmen who earn atmost 42000.



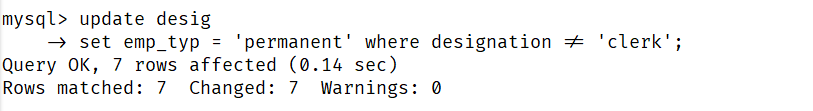
14) Add a record for ‘Analyst’ in desig table.



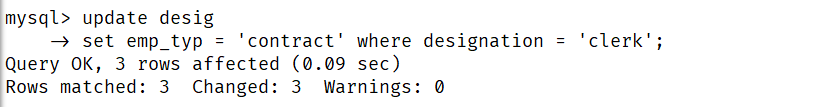
15) Add a column emp\_typ varchar(15) to desig.



16) Set the emp\_typ of all workers as ‘permanent’ except for ‘clerk’.

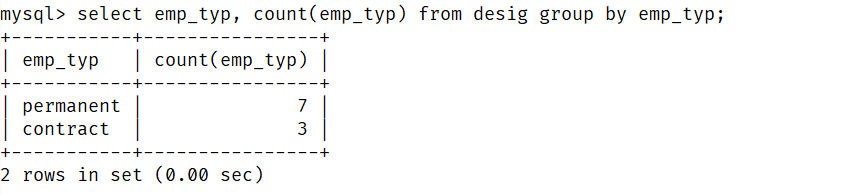


17) Set the emp\_typ of ‘clerk’ as ‘contract’.

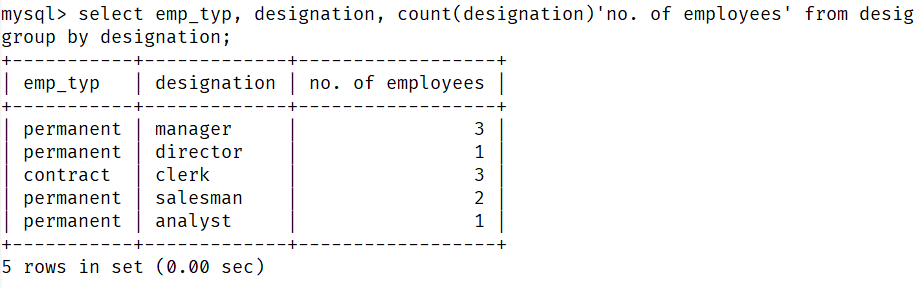


**40**

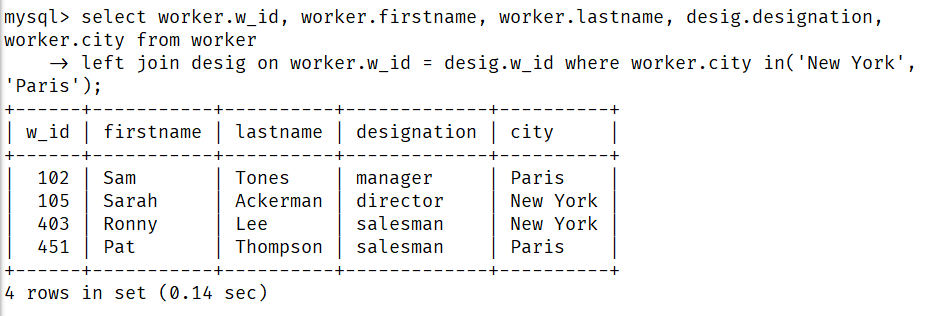
18) Count the number of workers of each type.



19) Count the number of employees for wach dsignation under emp\_type.

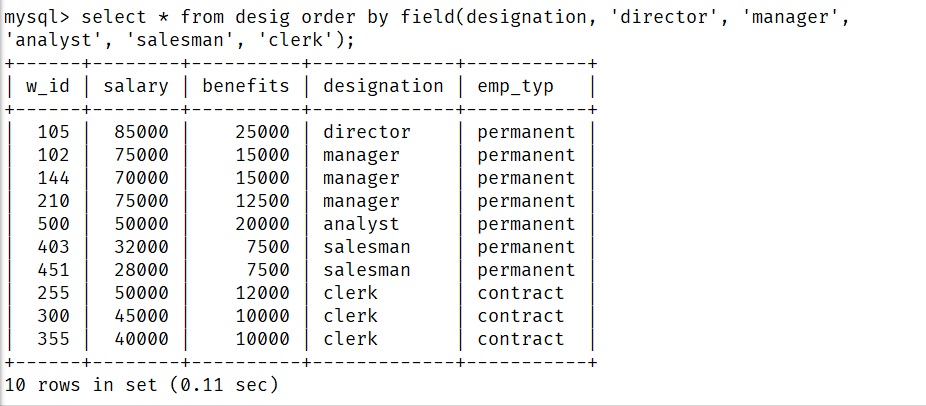


20) Show the w\_id, Firstname, Lastname, designation and city of each worker.

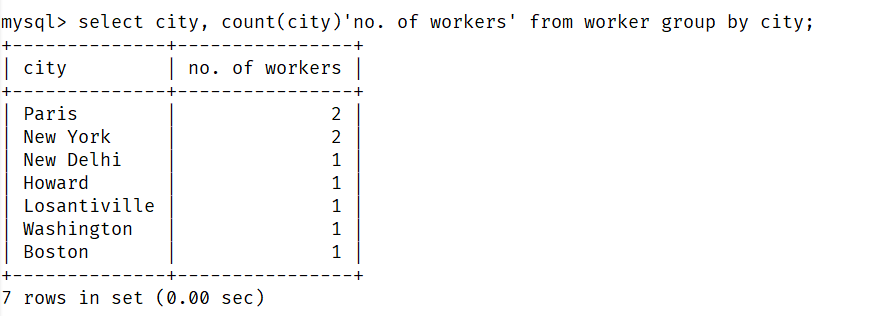


**41**

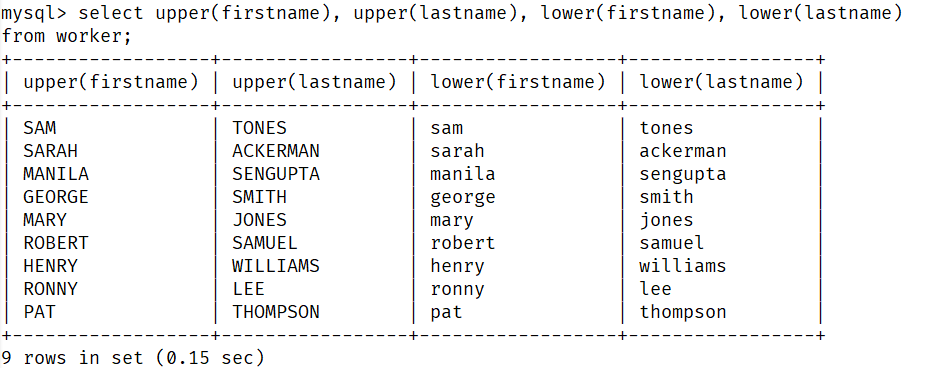
21) Display all designations in the order: Director, Manager, Analyst, Salesman, Clerk.



22) Display number of workers from each city.



23) Display Firstname and Lastname of each worker in upper case and lower case.



**42**