Task 9: Implement Exception and Ex cep

9.1 You are developing a Python program that
of students grades. The program is designed to allow the
user to select a grade by specifying an index number. However,
you need to ensure that the program handles cases where the user
inputs an index that is out of range.

Algorithm:
1. start
2. Initialize a list of grades
3. Prompts the user to enter the index of grade they wish to view
4. Attempts to display the grade at specified index.
5. If the index is out of range, catches the index Error and
Prints an error message "Invalid index.

Program:
# initialize the list of grades
grades = [85, 90, 78, 92, 88]
# Display the grades list
Print ("Grades List", grades)
Prompt the user to enter the index of the grade they want
to view
try:
index = int (input ("Enter the index of grade you want to view:"))
# Attempt to display the grade at the specified index.
Print (f"The grade at index {index} is {grades[index]}")
except IndexError:
Handle the case where the input is not an integer.
Print ("Invalid input. Please enter a numerical index.")

Output:
Grades List: [85, 90, 78, 92, 88]
Enter the index of grade you want to view: 10
Invalid Index. Please enter a valid index.

9.2 You are developing a Python calculator program that performs basic arithmetic operations. One of the key functions is to divide two numbers entered by the user. However, dividing by zero is not allowed and would cause Program to crash if not handled properly.

Algorithm:

1. Start the Program
2. Prompts the user to enter two numbers
3. Attempts to divide numerator by the denominator.
4. If the denominator is zero, catches the zero Division error and displays an error message.

Program:

```
# Function to Perform division
def divide_numbers():
    try:
# Prompt the user to enter the numerator.
        numerator = float(input("Enter the numerator:"))
# Prompt the user to enter the denominator:
        denominator = float(input("Enter the denominator:"))
# Attempt to perform division
        result = numerator / denominator
        Print(f"Result: {result}")
    except ZeroDivisionError:
# Handle division by zero error:
        Print("Error: Division by zero is not allowed.")
    except ValueError:
```

output:

```
Enter the numerator: 10
Enter the denominator: 0
Error!
Error: Division by zero is not allowed.
```

9.3. You are building a python application to determine if a person is eligible to vote based on their age. According to the rules, only individuals who are 18 years or older are allowed to vote. To enforce this rule, you decide to create a custom exception called invalidAge Exception which will be raised whenever an age below 18 is entered.

Algorithm:

1. Define the custom exception
2. Prompt the user for input.
3. Check if the age is below 18.
4. Raise an exception if condition is met.
5. Handle the exception with a custom error message.

Program:

```
# define Python user-defined exceptions
class invalidAge exception (Exception):
"Raised when the input value is less than 18"
  Pass
# you need to guess this number
  number = 18
try:
  input_num = int (input ("enter a number:"))
  if input_num < number:
  raise invalid Age exception
else:
  Print ("Eligible to vote")
except Invalid Age exception:
Printf ("exception occured: invalid Age")
```

Output:

Enter a number: 15
Exception occured: Invalid Age

VEL TECH - CSE	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	

Result:

Thus the program for implement Exception and Exceptional handling is executed and verified successfully.