Assignment-8(Exception Handling and File Handling)

1. Write a Python program that reads a file named data.txt. Use exception handling to manage the following scenarios:

- The file does not exist.
- The file is empty.
- Any unexpected error during file reading.

```
file = open(r"D:\BEBO TECHNOLOGY\python\Assignment-8(Exception Han-
dling and File Handling)\data.txt")

try:
    data = file.read()
    print(data)
except FileNotFoundError as e:
    print(e)
except ValueError as ve:
    print(f"Error: {ve}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

2. Create a program that accepts two numbers from the user and divides them. Handle the following exceptions:

- ZeroDivisionError if the divisor is zero.
- ValueError if the input is not a number.
- Any other generic exceptions.

```
try:
    num1 = int(input("Enter first number: "))
    num2 = int(input("ENter second number: "))
    result = num1//num2
    print(result)
except ZeroDivisionError as e:
    print(e)
except ValueError:
    print("Enter a numeric value")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

3. Write a function that opens a file, writes a list of numbers (1 to 10) into the file, and then closes it. Use a try-finally block to ensure the file is properly closed even if an exception occurs.

```
def write_numbers_to_file(numbers):
    try:
        file = open(r"D:\BEBO TECHNOLOGY\python\Assignment-8(Exception
Handling and File Handling)\numbers.txt", 'w')

        for number in range(1, 11):
            file.write(f"{number}\n")
        print(f"Numbers 1 to 10 have been written to {numbers}.")

    except Exception as e:
```

```
        print(f"An error occurred: {e}")

    finally:
        file.close()
        print("File has been closed.")


write_numbers_to_file("numbers.txt")
```

4. Write a script to read lines from a file records.txt and print them to the console. Add exception handling to manage file not found and permission errors.

```
try:
    file = open(r"D:\BEBO TECHNOLOGY\python\Assignment-8(Exception
Handling and File Handling)\records.txt")
    print(file.read())
except FileNotFoundError as e:
    print(e)
except PermissionError as e:
    print(e)
```

5. Create a Python program that takes a filename as input and writes user-provided data into it. Handle exceptions for:

- Invalid filename input (e.g., blank filename).
- Permission denied when writing to the file.

```
def write_to_file():
    try:
        file_name = input("Enter the filename to write data into:
").strip()

        if not file_name:
            raise ValueError("Filename cannot be blank.")

        file = open(file_name, 'w')
        data = input("Enter the data to write into the file: ")
        file.write(data)
        print(f"Data has been successfully written to {file_name}.")

    except ValueError as ve:
        print(f"Error: {ve}")

    except PermissionError:
        print(f"Error: Permission denied. Unable to write to the file
{file_name}.")

    except Exception as e:
        print(f"An unexpected error occurred: {e}")

    finally:
        try:
            file.close()
        except NameError:
            pass

write_to_file()
```

6. Write a function safe_divide(a, b) that performs division and raises a custom exception DivisionByZeroError if b is zero. Demonstrate its usage with appropriate exception handling.

```
class DivisionByZeroError(Exception):
    def __init__(self,message="Division by zero is not allowed"):
        super().__init__(message)
def safe_divide(a,b):
    if b==0:
        raise DivisionByZeroError()
    return a/b

try:
    a = int(input("Enter numerator (a): "))
    b = int(input("Enter denominator (b): "))

    result = safe_divide(a, b)
    print(f"Result: {result}")
except DivisionByZeroError as e:
    print(f"Error: {e}")

except ValueError:
    print("Error: Please enter valid numeric values.")
```

7. Create a script to copy content from one file to another. Handle exceptions for missing source file, permission errors, and other unforeseen issues.

```
def copy_file_content(source_file, destination_file):
    try:
        source = open(r"D:\BEBO TECHNOLOGY\python\Assignment-8(Excep-
tion Handling and File Handling)\file1.txt", 'r')
        destination = open(r"D:\BEBO TECHNOLOGY\python\Assignment-
8(Exception Handling and File Handling)\file2.txt", 'w')
        content = source.read()
        destination.write(content)
        print(f"Content successfully copied from {source_file} to
{destination_file}.")

    except FileNotFoundError:
        print(f"Error: The source file '{source_file}' does not ex-
ist.")

    except PermissionError:
        print(f"Error: Permission denied. Unable to write to the file
'{destination_file}'.")

    except Exception as e:
        print(f"An unexpected error occurred: {e}")

    finally:
        try:
            source.close()
        except NameError:
            pass
        try:
            destination.close()
        except NameError:
            pass
```

```
source_file = input("Enter the source file path: ")
destination_file = input("Enter the destination file path: ")
copy_file_content(source_file, destination_file)
```

8. Create a class FileProcessor with methods to:

- Read a file.
- Write to a file.
- Append data to a file.

```
class FileProcessor:
    def __init__(self, filename):
        self.filename = filename

    def read_file(self):
        try:
            with open(self.filename, 'r') as file:
                content = file.read()
                print(f"Content of {self.filename}:")
                print(content)
        except FileNotFoundError:
            print(f"Error: The file '{self.filename}' does not ex-
ist.")
        except PermissionError:
            print(f"Error: Permission denied when trying to read the
file '{self.filename}'.")
        except Exception as e:
            print(f"An unexpected error occurred while reading the
file: {e}")

    def write_file(self, content):
        try:
            with open(self.filename, 'w') as file:
                file.write(content)
                print(f"Content has been written to {self.filename}.")
        except PermissionError:
            print(f"Error: Permission denied when trying to write to
the file '{self.filename}'.")
        except Exception as e:
            print(f"An unexpected error occurred while writing to the
file: {e}")

    def append_to_file(self, content):
        try:
            with open(self.filename, 'a') as file:
                file.write(content)
                print(f"Content has been appended to {self.file-
name}.")
        except PermissionError:
            print(f"Error: Permission denied when trying to append to
the file '{self.filename}'.")
        except Exception as e:
            print(f"An unexpected error occurred while appending to
the file: {e}")

filename = input("Enter the filename: ")
processor = FileProcessor(filename)
```

```
processor.read_file()

content_to_write = input("Enter content to write to the file: ")
processor.write_file(content_to_write)

content_to_append = input("Enter content to append to the file: ")
processor.append_to_file(content_to_append)
```

Use exception handling in each method to manage issues like file not found, permission errors, and invalid input.

9. Write a program to merge the contents of two files, file1.txt and file2.txt, into a third file merged.txt. Handle all possible exceptions during file operations.

```
f1 =open(r"D:\BEBO TECHNOLOGY\python\Practice-Session-20(File Han-
dling)\file1.txt")
f2 =open(r"D:\BEBO TECHNOLOGY\python\Practice-Session-20(File Han-
dling)\file2.txt")
f3 =open(r"D:\BEBO TECHNOLOGY\python\Practice-Session-20(File Han-
dling)\mergefile.txt","w")

first = f1.read()
second = f2.read()
merge = first +" "+ second
f3.write(merge)
print("Merging successfully done.........")
f1.close()
f2.close()
f3.close()
```

10. Create a function process_file(file_path) that reads a file and returns the total number of words in it. Handle exceptions for missing file, empty file, and encoding errors.

```
def process_file(file_path):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
            if not content:
                print("The file is empty.")
                return 0
            words = content.split()
            return len(words)

    except FileNotFoundError:
        print(f"Error: The file '{file_path}' does not exist.")
        return 0

    except IOError:
        print(f"Error: There was an issue reading the file
'{file_path}'.")
        return 0

    except UnicodeDecodeError:
        print(f"Error: Encoding error while reading the file
'{file_path}'.")
        return 0
```

```
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return 0

file_path = input("Enter the path of the file: ")
word_count = process_file(file_path)
if word_count != 0:
    print(f"The total number of words in the file is: {word_count}")
```