Date _____ Page _____

UNIT - 3

Q2.

```
filename = "data.txt"

try:
    with open(filename, 'r') as file:
        for line in file:
            line = line.strip()
            try:
                number = int(line)
                print(f"Converted: {number}")
            except ValueError:
                print(f"Couldn't convert '{line}' to int")
                continue
except FileNotFoundError:
    print(f"File {filename} not found")
```

data.txt

```
42
123
Hello
456
```

Q2.

```
file_path="data2.text"

try:
    f= open(file_path, 'r')
    for each_line in f:
        each_line = each_line.strip()
        try:
            num = int(each_line)
            print(f"successfully converted : {num}")
        except ValueError:
            print(f"Error : '{each_line}' is not a
                valid integer")
            continue
    f.close()
except FileNotFoundError:
    print(f"Error: File '{file_path}' does not exist")
```

data.txt

```
50
200
test
999
```

Q3.

Q3. ~~out_file = "output.txt"~~

Q3. out_file = "output.txt"

user_input = input("Enter text to write to file")

try:
    with open(output_file, 'w') as file:
        file.write(user_input)
    print(f"Successfully wrote to {output_file}")
except PermissionError:
    print(f"Permission denied: cannot write to {output_file}")
except Exception as e:
    print(f"An error occured: {e}")

Q4. filename = input("Enter filename to create:")
content = input("Enter content to write")

try:
    f = open(filename, 'w')
    f.write(content)
    f.close()
    print(f"File '{filename}' created successfully")
except PermissionError:
    print(f"Error: No permission to create file '{filename}'")
except IOError as e:
    print(f"IO Error: {e}")

Q5.
```
text = input("Enter a string to convert to integer")
try:
    number = int(text)
    print(f"Conversion successful! no. is {number}")
except ValueError:
    print(f"Cannot convert '{text}' to an integer")
print("Program continued executing")
print("This line shows the program didn't stop")
```

Q6.
```
valid_marks = []
try:
    with open("marks.txt", 'r') as file:
        for line in file:
            try:
                valid_marks.append(int(line.strip()))
            except ValueError
                continue
    if valid_marks
        total = sum(valid_marks)
        print(f"Total : {total}")
        print(f"Average : {total} / (len(valid_marks):.2f}")
    else:
        print("No valid marks found")
except FileNotFoundError:
    print("File not found")
```

Q7.
```
in_file = "integer7_input.txt"
out_file = "integer7_output.txt"

try:
    with open(in_file) as fin:
        nums = []
        for not s:
            continue
        try:
            nums.append(int(s))
        except ValueError:
            pass
except FileNotFoundError:
    print("Input file not found")
else:
    try:
        with open(out_file, 'w') as fout:
            fout.write("\n".join(str(n) for n in nums))
        print(f"Saved {len(nums)} integer to '{out_file}'")
    except PermissionError:
        print("Permission denied for output file")
```

Q8.

```
in_file = "expenses_input.txt"
clean_file = "expenses_cleaned.txt"
rejected_file = "expenses_rejected.txt"

clean, rejected = [], []

try:
    with open(in_file) as fin:
        for raw in fin:
            line = raw.strip()
            if not line:
                rejected.append("Empty\t")
                continue
            parts = [p.strip() for p in line.split(",")]
            if len(parts) != 2 or not parts[0]:
                rejected.append(f"SPLIT OR CATEGORY\t{line}")
                continue
            cat, amt = parts
            try:
                val = float(amt)
            except ValueError:
                rejected.append(f"Amount Invalid\t{line}")
                continue
            if val < 0:
                rejected.append(f"Amount Negative\t{line}")
                continue
            clean.append(f"{cat.lower()},{val:.2f}")
```

except FileNotFoundError:
print("Input File not found")
else:
try:
with open(clean_file, "w") as cf:
cf.write("\n".join(clean))
with open(rejected_file, "w") as rf:
rf.write("\n".join(rejected))
print(f"cleaned {len(clean)}, Rejected {len(rejected)}")

except PermissionError:
print("Permission denied when writing")

Q9. import sys

F = "contacts.txt"
U = "options: \n search <term> \n add <name> <email> <phone> \n"

def search(t):
try:
with open(F) as f:
m = [c.strip() for c in f if t.lower() in c.lower()]
except FileNotFoundError:
return print(f"missing file: {F}")
print("matches:", m if m else "No matches found")
for x in m: print(x)

```
def add(n, e, p):
    try:
        with open(F, "a") as f: f.write(f"{n},{e},{p}\n")
        print("contact added")
    except Exception as ex:
        print(f"write error {ex}")

def main(a):
    if len(a) < 2: return print("Invalid arg")
    c = a[1]
    if c == "search":
        return print("Provide term") if len(a) < 3
        else search(a[2])
    if c == "add":
        return print("Provide name email phone")
        if len(a) < 5 else add(*a[2:5])
    print("Invalid option\n" + U)

if __name__ == "__main__":
    main(sys.argv)
```

Q10.

```
class InvalidNumberError(Exception): pass

file = "numbers.txt"
valid = []

try:
    for line in open(file):
        s = line.strip()
        if not s: continue

        if any(c.isalpha() for c in s):
            print(f"Letters found in line: {s}")
            continue

        try:
            valid.append(int(s))
        except ValueError:
            pass

except FileNotFoundError:
    print("Input file not found")
else:
    print(f"Collected {len(valid)} valid integers: {valid}")
```

Q10

```
class NegativeMarkError(Exception): pass
class HighMarkError(Exception): pass

in_file = "marks.txt"
out_file = "clean_marks.txt"
valid = []

try:
    with open(in_file) as f:
        for raw in f:
            s = raw.strip()
            if not s: continue

            try:
                m = int(s)
                if m < 0: raise NegativeMarkError(f"Neg...")
                if m > 100: raise HighMarkError(f"mark above")
                valid.append(m)
            except ValueError:
                print(f"Invalid no")
    except (NegativeMarkError, HighMarkError) as e:
        print(e)
```