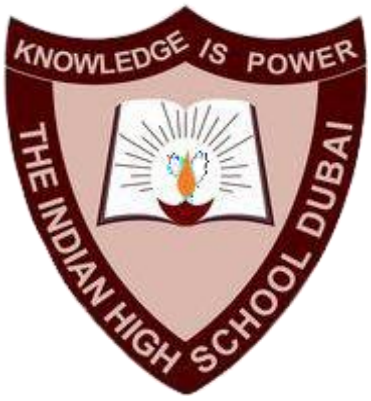
THE INDIAN HIGH SCHOOL - DUBAI



Comp. Sci Journal
2025-26

Name: Abyaz Javid
Roll no.: 4

CERTIFICATE

This is to certify that the work in this journal is the bonafide work of

Master _____

Class _____ *Div* _____

recorded in the school lab during the academic year

20xx - 20xx

Date: _____

Teacher in charge: _____

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the Central Board of Secondary Education (CBSE) and The Indian High School-Dubai, for granting me the opportunity to deepen my knowledge in my favorite subject, Computer Science

I would also like to thank my teacher Mrs. Swapnil Verma for guiding me and sharing her wide variety of knowledge

INDEX

FUNCTIONS

Program 1

Aim: To write a function that takes 'n' number of integers and count the number of odd and even numbers.

Modules used: N/A

Data types used: Int

Script:

```
def count(*a) -> None:
    o, e = 0, 0
    for i in a:
        (e := e + 1) if i % 2 == 0 else (o := o + 1)
    print(f"No. of odd elements: {o}\nNo. of even elements: {e}")

count(1,2,3,4,5,6,7,8)
```

Output:

```
No. of odd elements: 4
No. of even elements: 4
>>>
```

Program 2

Aim: To write a function to find the sum of the series: $1 + x^2 + x^3 + \cdots + x^n$

Modules used: N/A

Data types used: Int

Script:

```
def series(x: int, n: int) -> None:
    s = 1
    for i in range(n): s += x**(i+1)
    print(f"Sum of the series is: {s}")

series(2, 16)
```

Output:

```
    Sum of the series is: 131071
>>>
```

Program 3

Aim: To write a function to find the factorial of a number without taking an argument

Modules used: N/A

Data types used: Int

Script:

```
n = int(input(">>> "))
def fact() -> int:
    global n
    return n * (n := n - 1, fact())[1] if n > 1 else 1

print(f"Factorial of {n} is: {fact()}")
```

Output:

```
>>> 5
Factorial of 5 is: 120
```

Program 4

Aim: To write a function that takes a list of strings and return the emails that contain the substring "@cmail"

Modules used: N/A

Data types used: String, List

Script:

```
r = input(">>> ").split()
def validMailID(l: list) -> list:
    return [i for i in l if "@cmail" in i]

print(f"Valid email IDs are:\n - " + "\n - ".join(validMailID(r)))
```

Output:

```
>>> asdf@cmail.com fdsa@cmail.com asdf@gmail.com swag@yahoo.gov
Valid email IDs are:
 - asdf@cmail.com
 - fdsa@cmail.com
```

Program 5

Aim: To write a function that takes a list of strings and returns the strings that are longer than 5 characters.

Modules used: N/A

Data types used: String, List

Script:

```
r = input(">>> ").split()
def longWords(l: list):
    return [i for i in l if len(i) > 5]

print(f"Words longer than 5 letters are:\n - " + "\n - ".join(longWords(r)))
```

Output:

```
>>> word1 antidisestablishmentarianism word2 wee asdf fdsa
Words longer than 5 letters are:
 - antidisestablishmentarianism
```

Program 6

Aim: To write a menu driven program to find odd/even numbers and prime numbers.

Modules used: math

Data types used: Int, Bool

Script:

```
import math

print("""
    #--------------------#
    |       NUMBERS      |
    | 1. even/odd        |
    | 2. prime/consonant |
    | 3. exit            |
    #--------------------#""")

def evenity(n: int) -> bool:
    return True if n % 2 == 0 else False

def primality(n: int) -> bool:
    return False if n <= 1 else (True if n == 2 else (False if n % 2 == 0 else all(n % i != 0 for i in range(3, int(math.sqrt(n)) + 1, 2))))


while True:
    o = int(input(">>> "))
    if o == 1:
        print(evenity(int(input("n: "))))
    elif o == 2:
        print(primality(int(input("n: "))))
    elif o == 3:
        break
    else:
        print("Invalid option selected")
```

Output:

```
    #--------------------#
    |       NUMBERS      |
    | 1. even/odd        |
    | 2. prime/consonant |
    | 3. exit            |
    #--------------------#
>>> 1
n: 43
False
>>> 2
n: 5
True
>>> 4
Invalid option selected
>>> 3
```

Program 7

Aim: To write a function that returns the greater of two numbers.

Modules used: N/A

Data types used: Tuple, Float

Script:

```
a, b = tuple([*map(float, input(">>> ").split())])
def findBig() -> float:
    global a, b
    return a if a > b else b

print(f"Bigger number: {findBig()}")
```

Output:

```
>>> 1 3
Bigger number: 3.0
```

Program 8

Aim: To write a function that takes a list and moves all the elements divisible by 5 to the end of the list.

Modules used: N/A

Data types used: List, Int

Script:

```
x = [*map(int, input(">>> ").split())]
def move(l: list) -> None:
    l[:] = [x for x in l if x % 5] + [x for x in l if x % 5 == 0]
    print(f"Ordered list is: {l}")

move(x)
```

Output:

```
>>> 1 2 5 4 6 58 65 2350 15
Ordered list is: [1, 2, 4, 6, 58, 5, 65, 2350, 15]
```

Program 9

Aim: Given a dictionary containing information about vehicles, display the vehicles that were released in 2020 and order the dict in alphabetical order by brand name.

Modules used: N/A

Data types used: Dict, Int, List, String

Script:

```
vehicle = {
    "car1": ["Toyota", "Camry", 2020, 25_000],
    "car2": ["Ford", "Explorer", 2019, 32_000],
    "car3": ["Chevy", "Silverado", 2021, 40_000],
    "car4.5": ["Honda", "Civic", 2020, 22_000],
    "car5": ["anotherRealCarBrand", "Model nine", 2023, 45_000]
}

def _2020(D: dict):
    print(f"No. of vehicles released in 2020: {len([c for c in D if c[2] == 2020])}")

def sort(cars_dict):
    sorted_cars = sorted(cars_dict.items(), key=lambda x: x[1][0].lower())
    for key, value in sorted_cars:
        print(f"{key}: {value}")

_2020(vehicle)
print("\nSorted dict:")
sort(vehicle)
```

Output:

```
No. of vehicles released in 2020: 0

Sorted dict:
car5: ['anotherRealCarBrand', 'Model nine', 2023, 45000]
car3: ['Chevy', 'Silverado', 2021, 40000]
car2: ['Ford', 'Explorer', 2019, 32000]
car4.5: ['Honda', 'Civic', 2020, 22000]
car1: ['Toyota', 'Camry', 2020, 25000]
```

Program 10

Aim: To write a function that takes a tuple and returns the indices of the non-zero elements.

Modules used: N/A

Data types used: Int, Tuple, List

Script:

```
t = tuple([*map(int, input(">>> ").split())])
def indexTuple(t: tuple) -> list:
    return [i for i, v in enumerate(t) if v != 0]

print(f"Non zero indices are: {indexTuple(t)}")
```

Output:

```
>>> 1 0 25 03 64 00 5
Non zero indices are: [0, 2, 3, 4, 6]
```

Program 11

Aim: To write a function to count the number of vowels in user input.

Modules used: N/A

Data types used: String

Script:

```
r = input(">>> ")
def vowelCount() -> None:
    global r
    print(f"No. of vowels: {sum(i in "AEIOUaeiou" for i in r)}")

vowelCount()
```

Output:

```
>>> hello my name is star.stalker9160
No. of vowels: 8
```

FILE HANDLING – 1

TEXT FILES

Program 1

Aim: Write a function to count the number of lines that start with the alphabet 'W' or 'H'

Modules used: N/A

Data types used: Int, Str

Script:

```
def f() -> None:
    with open('dump/Journal Files/Country.txt') as f: return sum(1 for i in f.readlines() if i[0] in 'WH')

print(f"No. of words starting with W or H: {f()}")
```

Output:

```
No. of words starting with W or H: 1
>>>
```

Program 2

Aim: Write a function countWords() to display total number of words in a file

Modules used: N/A

Data types used: Int, Str

Script:

```
def countWords() -> None:
    with open('dump/Journal Files/Quotes.txt') as f: print(f"No. of words: {len([x for x in f.read().split() if x != "\n"])}")

countWords()
```

Output:

```
------------------
No. of words: 6
>>>
```

Program 3

Aim: Write a function filter(oldfile, newfile) that copies all lines from oldfile into newfile that don't start with '@'

Modules used: N/A

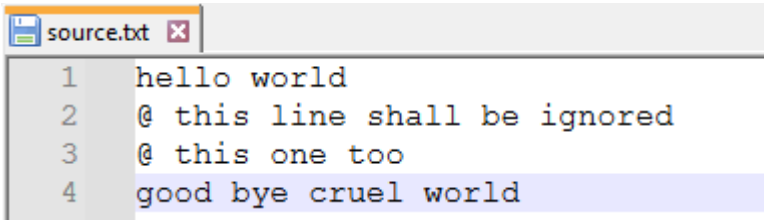
Data types used: Int, Str

Script:

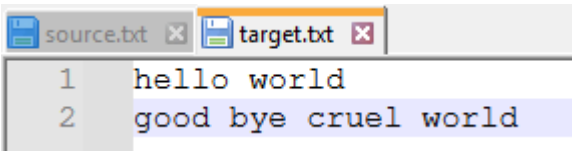
```
def filter(oldfile: str, newfile: str) -> None:
    with open(oldfile) as o, open(newfile, "w") as n:
        n.writelines([l for l in o if not l.startswith("@")])

filter("dump/Journal Files/source.txt", "dump/Journal Files/target.txt")
```

Output:

source.txt:



target.txt



Program 4

Aim: Write a function VowelCount that displays the occurrence of vowels in a file

Modules used: json

Data types used: Int, Str, Dict

Script:

```
import json

def VowelCount() -> None:
    with open("dump/Journal Files/MY_TEXT_FILE.txt") as f:
        t = f.read().lower()
    print(json.dumps({v: t.count(v) for v in 'aeiou'}, indent=4))

VowelCount()
```

Output:

```
{
    "a": 1,
    "e": 2,
    "i": 5,
    "o": 4,
    "u": 0
}
>>>
```

Program 5

Aim: Write a function to count the occurrence of ‘The’ and ‘This’

Modules used: N/A

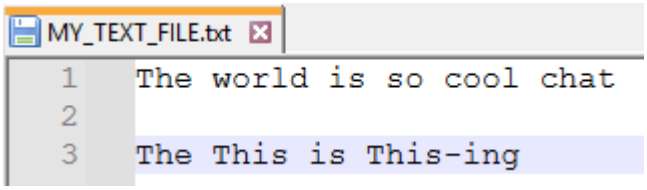
Data types used: Int

Script:

python:

```
def f() -> None:
    with open('dump/Journal Files/MY_TEXT_FILE.txt') as f: return sum(1 for i in f.read().split() if (i == "The" or i == "This"))

print(f"Occourance of 'the' or 'this': {f()}")
```

MY_TEXT_FILE.txt:



Output:

```
>>> Occourance of 'the' or 'this': 3
```

Program 6

Aim: Write a function ISTOUPCOUNT to count the occurrence of 'IS', 'TO' and 'UP' in a file

Modules used: json

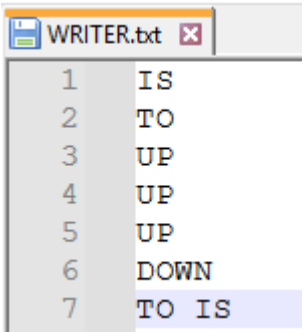
Data types used: Str, Dict, Int

Script:

python:

```
import json

def ISTOUPCOUNT() -> None:
    with open("dump/Journal Files/WRITER.txt") as f:
        w = f.read().split()
        print(json.dumps({"IS": w.count("IS"), "TO": w.count("TO"), "UP": w.count("UP")}, indent=4))

ISTOUPCOUNT()
```

WRITER.txt:



Output:

```
{
    "IS": 2,
    "TO": 2,
    "UP": 3
}
>>>
```

Program 7

Aim: Write a function to print out the lines from a file that start with 'P'

Modules used: N/A

Data types used: Str, List

Script:

python:

```
def p() -> None:
    with open("dump/Journal Files/DIARY.txt") as f:
        [print(line, end="") for line in f if line.startswith("P")]

p()
```

DIARY.txt:

DIARY.txt

```
1   P line number 1
2   line that does not start with p
3   P <- heres another p line
```

Output:

```
P line number 1
P <- heres another p line
>>>
```

Program 8

Aim: Write a function to display the number of lines starting with 'H'

Modules used: N/A

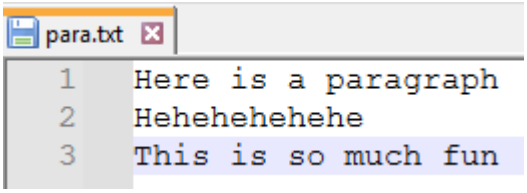
Data types used: Int

Script:

python:

```
def h() -> None:
    with open("dump/Journal Files/para.txt") as f:
        print(sum(1 for l in f if l.startswith("H")))

h()
```

para.txt:



Output:

```
2
>>>
```

Program 9

Aim: Write a function AMCount to count the occurrences of 'a' and 'm' both upper and lower case

Modules used: json

Data types used: Str, Int, Dict

Script:

python:

```
import json

def AMCount() -> None:
    with open("dump/Journal Files/STORY.txt") as f:
        t = f.read()
        print(json.dumps({'a': t.count('a'), 'A': t.count('A'), 'm': t.count('m'), 'M': t.count('M')}, indent=4))

AMCount()
```

STORY.txt:

STORY.txt

```
1  i am wake up at 4am in the marning
2  and like
3  and
4  uhhhhhhh
5  idk
```

Output:

```
{
    "a": 7,
    "A": 0,
    "m": 3,
    "M": 0
}
>>>
```

Program 10

Aim: Write a function COUNT to count the occurrence of 'Catholic' and 'mother'

Modules used: json

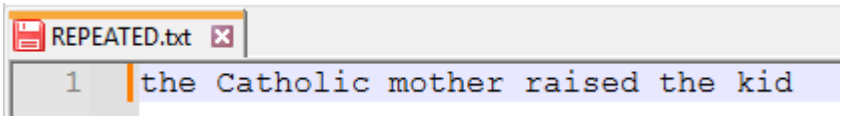
Data types used: Int, Str, Dict

Script:

python:

```
import json

def COUNT():
    with open("dump/Journal Files/REPEATED.txt") as f:
        t = f.read().split()
        print(json.dumps({"Catholic": t.count("Catholic"), "mother": t.count("mother")}, indent=4))

COUNT()
```

REPEATED.txt:



Output:

```
{
    "Catholic": 1,
    "mother": 1
}
>>>
```

Program 11

Aim: Write a function to print out the lines that have only 2 chars

Modules used: N/A

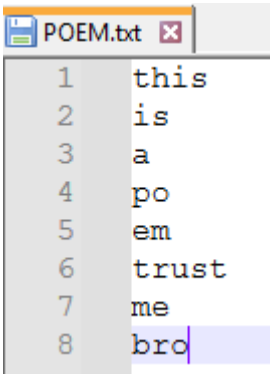
Data types used: Int, List, Str

Script:

python:

```
def _2chars():
    with open("dump/Journal Files/POEM.txt") as f:
        [print(w) for l in f for w in l.split() if len(w) == 2]

_2chars()
```

POEM.txt:



Output:

```
is
po
em
me
>>>
```

Program 12

Aim: Write a function COUNT_AND to count the occurrence of ‘and’ (case insensitive)

Modules used: N/A

Data types used: Int, Str

Script:

python:

```
def COUNT_AND():
    with open("dump/Journal Files/STORY.txt") as f:
        text = f.read().lower()
        print("Occourance of 'and': ", text.count("and"))

COUNT_AND()
```

STORY.txt:

STORY.txt

```
1  i am wake up at 4am in the marning
2  and like
3  and
4  uhhhhhhh
5  idk
```

Output:

```
>>> Occourance of 'and':  2
```

Program 13

Aim: Write a menu driven program to perform operations on a file

Modules used: N/A

Data types used: Int, List, Str

Script:

```
_f = "dump/Journal Files/POETIC.txt"

def CREATE():
    with open(_f, "w") as f:
        for i in range(int(input("Enter number of lines: "))):
            f.write(input(f"{i+1}. ") + "\n")

def DISPLAY():
    with open(_f) as f:
        print(f.read())

def COUNTCHAR():
    c = {"v": 0, "c": 0, "u": 0, "l": 0}
    with open(_f) as f:
        for char in f.read():
            if char in "aeiouAEIOU": c["v"] += 1
            elif char in "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ": c["c"] += 1
            if char.isupper(): c["u"] += 1
            elif char.islower(): c["l"] += 1
    print(c)

def HASHSHOW():
    with open(_f) as f:
        for l in f: print("#".join(l.split()))

def COPY():
    with open(_f) as f:
        l = f.readlines()
    with open("dump/Journal Files/special.txt", "w") as f1:
        f1.writelines([i for i in l if "#" in i])

def REPLACE():
    st, r = input("search: "), input("replace: ")
    with open(_f) as f:
        t = f.read()
    ct = t.replace(st, r)
    with open("dump/Journal Files/duplicate.txt", "w") as f:
        f.write(ct)
    print({"Original text": t, "Changed text": ct})

def DELETE():
    w = input("Enter the word to delete: ")
    with open(_f) as f:
        text = f.read()
    with open(_f, "w") as f:
        f.write(text.replace(w, ""))

def COUNTEND():
    with open(_f) as f:
        print({"count": sum(1 for line in f if line.rstrip().endswith(("y", "i")))})

def VOWEL():
    with open(_f) as f:
        t = f.read()
    with open("dump/Journal Files/vowel.txt", "w") as v:
        v.writelines([w + "\n" for w in t.split() if w[0].lower() in "aeiou"])
    print({"Original file": t, "Vowel file": open("dump/Journal Files/vowel.txt").read()})

def CHANGE():
    with open(_f) as f:
        text = f.read()
    ct = text.replace(" ", "**")
    with open("dump/Journal Files/changed.txt", "w") as f:
        f.write(ct)
    print({"Original text": text, "Changed text": ct})
```

```
print("""
        #-----------------------#
        |          rEee         |
        |   1. create           |
        |   2. display          |
        |   3. count characters |
        |   4. hash show        |
        |   5. copy             |
        |   6. replace          |
        |   7. delete           |
        |   8. count end        |
        |   9. vowel            |
        |   10. change          |
        |   11. exit            |
        |                       |
        #-----------------------#""")

options = {1: CREATE, 2: DISPLAY, 3: COUNTCHAR, 4: HASHSHOW, 5: COPY, 6: REPLACE, 7: DELETE, 8: COUNTEND, 9: VOWEL, 10: CHANGE}

while True:
    o = int(input(">>>"))
    if o == 11:
        break
    elif o in options:
        options[o]()
    else:
        print("Invalid option")
```

Output:

```
        #-----------------------#
        |          rEee         |
        |   1. create           |
        |   2. display          |
        |   3. count characters |
        |   4. hash show        |
        |   5. copy             |
        |   6. replace          |
        |   7. delete           |
        |   8. count end        |
        |   9. vowel            |
        |  10. change           |
        |  11. exit             |
        |                       |
        #-----------------------#
>>>1
Enter number of lines: 2
1. asdf
2. fdsa
>>>2
asdf
fdsa

>>>3
{'v': 2, 'c': 6, 'u': 0, 'l': 8}
>>>4
asdf
fdsa
>>>5
>>>6
search: asdf
replace: qwre
{'Original text': 'asdf\nfdsa\n', 'Changed text': 'qwre\nfdsa\n'}
>>>7
Enter the word to delete: asdf
>>>2

fdsa

>>>8
{'count': 0}
>>>9
{'Original file': '\nfdsa\n', 'Vowel file': ''}
>>>10
{'Original text': '\nfdsa\n', 'Changed text': '\nfdsa\n'}
>>>11
>>>
```