System Security Lab-1

Praneesh R V

CB.SC.U4CYS23036

Q1. Create a Makefile to print Hello World on the terminal.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

hello:
        echo "Hello World"
~
~
~
~
```

```
crimsonshadow@L:~/SystemSecurity$ ls
Makefile
crimsonshadow@L:~/SystemSecurity$
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity$ make
echo "Hello World"
Hello World
crimsonshadow@L:~/SystemSecurity$
```

Q2. Create a Makefile with two targets.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

file1 : file2
                echo "This will always run, and runs second"
file2:
        echo "This will always run, and runs first"
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity$ ls
Makefile
crimsonshadow@L:~/SystemSecurity$
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity$ make
echo "This will always run, and runs first"
This will always run, and runs first
echo "This will always run, and runs second"
This will always run, and runs second
crimsonshadow@L:~/SystemSecurity$
```

Q3. Create a calculator program using Makefile. (Eg. calculator.h that has add.c, subtract.c, multiply.c, divide.c, other.c and a main calculator.c)

Makefile

```
CC = gcc
CFLAGS = -Wall -g

all: main
main: main.o calc.o
        $(CC) $(CFLAGS) -o main main.o calc.o
main.o: main.c calculator.h
        $(CC) $(CFLAGS) -c main.c

calculator.o: calculator.c calculator.hl
        $(cc) $(CFLAGS) -c calc.c
clean:
        rm -f *.o main
~
```

Calc.c

```c
#include "calculator.h"
int add(int a, int b){
return a+b;
}
int sub(int a, int b){
return a-b;
}
int multiply(int a, int b){
return a*b;
}
float divide(float a, float b){
return a / b;
} else {
return 0;
}
int other(int a, int b){
return a%b;
}
~
~
~
calc.c
```

Calculator.h

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

#ifndef CALCULATOR_H
#define CALCULATOR_H

int add(int a, int b);
int sub(int a, int b);
int multiply(int a, int b);
float divide(float a, float b);
int other(int a, int b);

#endif

~
~
~
~
~
~
~
~
~
calculator.h [+]
```

Main.c

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

#include <stdio.h>
#include <calculator.h>

int main(){
    int a=100,b=50;
    printf("Addition=%d \n",add(a,b));
    printf("Subtraction=%d \n",sub(a,b));
    printf("Multiplication=%d \n",multiply(a,b));
    printf("Division=%f \n",divide(a,b));
    printf("Modulus=%d \n",other(a,b));
    return 0;
}
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity$ ls
Makefile  calc.c  calculator.h  main.c  main.o
crimsonshadow@L:~/SystemSecurity$ make
gcc -Wall -g   -c -o calc.o calc.c
gcc -Wall -g -o main main.o calc.o
crimsonshadow@L:~/SystemSecurity$ ls
Makefile  calc.c  calc.o  calculator.h  main  main.c  main.o
crimsonshadow@L:~/SystemSecurity$ ./main
Addition=150
Subtraction=50
Multiplication=5000
Division=2.000000
Modulus=0
crimsonshadow@L:~/SystemSecurity$
```

Q4. Create a simple Python program to fetch random trivia about a user given number. Create Makefile to run the app.

Req.txt

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

requests

~
~
~
~
~
~
~
```

App.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

import api
api.get_fact(input("Enter a number: "))
```

Api.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

import requests
def get_fact(number):
    url = "http://numbersapi.com/{}".format(number)
    r = requests.get(url)
    if r.status_code == 200:
        print(r.text)
    else:
        print("An error occurred,code={}".format(r.status_code))
```

Makefile

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

clean:
        rm -rf __pycache__
run:
        python app.py
setup: req.txt
        pip install -r req.txt
~
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity/Qn4$ ls
Makefile  api.py  app.py  req.txt
crimsonshadow@L:~/SystemSecurity/Qn4$ make run
python3 app.py
Enter a number: 13
13 is the number of Oscar nominations of actress Meryl Streep, who holds the record for the most Oscar nominated actress.
crimsonshadow@L:~/SystemSecurity/Qn4$ ls
Makefile  __pycache__  api.py  app.py  req.txt
crimsonshadow@L:~/SystemSecurity/Qn4$ make clean
rm -rf __pycache__
crimsonshadow@L:~/SystemSecurity/Qn4$ ls
Makefile  api.py  app.py  req.txt
crimsonshadow@L:~/SystemSecurity/Qn4$
```

Qn5. . Create a Python program to count the number of objects in an image. Create a Python program to count the number of faces in an image. Create a third program to cartoon an image. Create a Makefile to choose among these and provide the required image as result.

Makefile

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

noi :
        python3 noi.py
nof :
        python3 nof.py
cartoon:
        python3 cartoon.py
setup:
        pip install -r req.txt
```

Req.txt

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

opencv-python
numpy
matplotlib
~
~
```

Noi.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

import cv2

def count_objects(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    _, threshold = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

    # Find contours
    contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Count the contours
    object_count = len(contours)
    print(f"Number of objects detected: {object_count}")

    # Visualize contours
    image_colored = cv2.imread(image_path)
    cv2.drawContours(image_colored, contours, -1, (0, 255, 0), 2)

    # Save the output image
    cv2.imwrite("output.jpg", image_colored)
    print("Output saved as output.jpg")
```

Nof.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
import cv2

def count_faces(image_path):
    # Load the Haar Cascade for face detection
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

    # Load the image
    image = cv2.imread(image_path)
    if image is None:
        print("Error: Could not read the image.")
        return

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    # Count the faces
    face_count = len(faces)
    print(f"Number of faces detected: {face_count}")
```

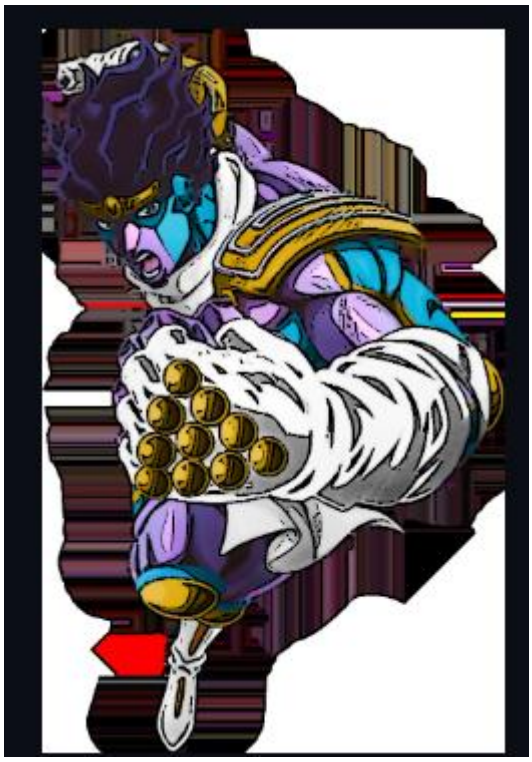
Cartoon.py

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
import cv2

def cartoonize_image(image_path):
    # Load the image
    image = cv2.imread(image_path)

    # Convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply median blur
    gray_blurred = cv2.medianBlur(gray_image, 5)

    # Detect edges
    edges = cv2.adaptiveThreshold(gray_blurred, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 9)

    # Apply bilateral filter to smooth colors
    color = cv2.bilateralFilter(image, 9, 300, 300)

    # Combine edges with the color image
    cartoon = cv2.bitwise_and(color, color, mask=edges)

cartoon.py
```

Output:

cartoon

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity/Lab1/Qn5$ make cartoon
python3 cartoon.py
Cartoonized image saved as: cartoonized_image.jpg
crimsonshadow@L:~/SystemSecurity/Lab1/Qn5$
```



Noi

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

crimsonshadow@L:~/SystemSecurity/Lab1/Qn5$ make noi
python3 noi.py
Number of objects detected: 228
Output saved as output.jpg
crimsonshadow@L:~/SystemSecurity/Lab1/Qn5$
```