**1.Boolean Operations**

#include <stdio.h>

// Function to perform the **AND** operation

int AND(int a, int b) {

return (a && b);

}

// Function to perform the **OR** operation

int OR(int a, int b) {

return (a || b);

}

// Function to perform the **NOT** operation

int NOT(int a) {

return (!a);

}

// Function to perform the **XOR** operation

int XOR(int a, int b) {

return ((a && !b) || (!a && b));

}

// Function to perform the **NAND** operation

int NAND(int a, int b) {

return !(a && b);

}

int **NOR**(int a, int b) {

return !(a || b);

}

int main() {

int a = 1, b = 0;

printf("AND(%d, %d) = %d\n", a, b, AND(a, b));

printf("OR(%d, %d) = %d\n", a, b, OR(a, b));

printf("NOT(%d) = %d\n", a, NOT(a));

printf("XOR(%d, %d) = %d\n", a, b, XOR(a, b));

printf("NAND(%d, %d) = %d\n", a, b, NAND(a, b));

printf("NOR(%d, %d) = %d\n", a, b, NOR(a, b));

return 0;

}

**2.Program to Grade Marks**

Csv file

def Calculate\_grade(marks):

if 0 <= marks < 40:

return 'E'

elif 40 <= marks < 50:

return 'D'

elif 50 <= marks < 60:

return 'C'

elif 60 <= marks < 70:

return 'B'

elif 70 <= marks <= 100:

return 'A'

else:

return 'Invalid Marks'

def Calculate\_average\_and\_grade(marks\_list):

average\_marks = sum(marks\_list) / len(marks\_list)

average\_grade = Calculate\_grade(average\_marks)

return average\_marks, average\_grade

def main():

student\_name = input("Enter the student's name: ")

registration\_number = input("Enter the student's registration number: ")

units = []

marks = []

grades = []

for i in range(1, 4):

unit\_name = input(f"Enter the name of unit {i}: ")

unit\_marks = int(input(f"Enter the marks for {unit\_name}: "))

units.append(unit\_name)

marks.append(unit\_marks)

grades.append(Calculate\_grade(unit\_marks))

average\_marks, average\_grade = Calculate\_average\_and\_grade(marks)

csv\_data = [

["Student Name", student\_name],

["Registration Number", registration\_number]

]

for i in range(3):

csv\_data.append([units[i], grades[i]])

csv\_data.append(["Average Marks", average\_marks, average\_grade])

# Write to CSV file

csv\_filename = f"{student\_name}\_grades.csv"

with open(csv\_filename, mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(csv\_data)

print(f"\nStudent details and grades have been saved to {csv\_filename}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3.Parameter Passing**

#include <stdio.h>

#include <stdlib.h>

// Function to demonstrate immutability (passing by value)

void demonstrate\_immutable(int value) {

printf("Inside function before change: %d\n", value);

value += 10;

printf("Inside function after change: %d\n", value);

}

// Function to demonstrate mutability (passing by reference)

void demonstrate\_mutable(int\* mutable\_list, int size) {

printf("Inside function before change: ");

for (int i = 0; i < size; ++i) {

printf("%d ", mutable\_list[i]);

}

printf("\n");

// Append 100 to the array

mutable\_list[size] = 100;

size += 1; // Update the size to reflect the new element

printf("Inside function after change: ");

for (int i = 0; i < size; ++i) {

printf("%d ", mutable\_list[i]);

}

printf("\n");

}

// Function to demonstrate function references

void demonstrate\_reference() {

printf("Inside function\n");

}

// Function to demonstrate various features of a function (simplified for C)

void function\_features(int param1, int param2) {

printf("param1: %d\n", param1);

printf("param2: %d\n", param2);

}

int main() {

// Pass by value (immutable type)

int immutable\_value = 5;

printf("Pass by Value:\n");

printf("Before function call: %d\n", immutable\_value);

demonstrate\_immutable(immutable\_value);

printf("After function call: %d\n", immutable\_value);

printf("----------------------------------------\n");

// Pass by reference (mutable type)

int mutable\_list[4] = {1, 2, 3}; // Allocate an extra element for the new value

int size = 3; // Current size of the array

printf("Pass by Variable (Mutable):\n");

printf("Before function call: ");

for (int i = 0; i < size; ++i) {

printf("%d ", mutable\_list[i]);

}

printf("\n");

demonstrate\_mutable(mutable\_list, size);

size += 1; // Update size to reflect the change in the function

printf("After function call: ");

for (int i = 0; i < size; ++i) {

printf("%d ", mutable\_list[i]);

}

printf("\n");

printf("----------------------------------------\n");

// Pass by reference

printf("Pass by Reference:\n");

void (\*function\_reference)() = demonstrate\_reference;

printf("Function reference: %p\n", (void\*)function\_reference);

function\_reference();

printf("----------------------------------------\n");

// Demonstrating various features of a function

printf("Function Features:\n");

function\_features(10, 20);

function\_features(10, 30);

return 0;

}