Level 2:

1)def employee\_bonus():

"""Program to calculate the bonus of 10 employees based on years of service."""

num\_employees = 10

salaries = []

years\_of\_service = []

bonuses = []

new\_salaries = []

total\_bonus = 0

total\_old\_salary = 0

total\_new\_salary = 0

for i in range(num\_employees):

while True:

try:

salary = float(input(f"Enter salary for employee {i+1}: "))

years = float(input(f"Enter years of service for employee {i+1}: "))

if salary <= 0 or years < 0:

print("Invalid input, please enter again.")

continue

break

except ValueError:

print("Invalid input, please enter a number.")

bonus = 0.05 \* salary if years > 5 else 0.02 \* salary

new\_salary = salary + bonus

salaries.append(salary)

years\_of\_service.append(years)

bonuses.append(bonus)

new\_salaries.append(new\_salary)

total\_bonus += bonus

total\_old\_salary += salary

total\_new\_salary += new\_salary

print("\nTotal Bonus Payout:", total\_bonus)

print("Total Old Salary:", total\_old\_salary)

print("Total New Salary:", total\_new\_salary)

2)def youngest\_and\_tallest():

"""Program to find the youngest and tallest among Amar, Akbar, and Anthony."""

names = ["Amar", "Akbar", "Anthony"]

ages = []

heights = []

for name in names:

age = int(input(f"Enter age of {name}: "))

height = float(input(f"Enter height of {name} in cm: "))

ages.append(age)

heights.append(height)

youngest = names[ages.index(min(ages))]

tallest = names[heights.index(max(heights))]

print(f"Youngest friend: {youngest}")

print(f"Tallest friend: {tallest}")

3)def find\_largest\_numbers():

"""Program to find the largest and second largest digits of a number."""

number = int(input("Enter a number: "))

digits = []

max\_digit = 10

index = 0

while number > 0:

digit = number % 10

if index == max\_digit:

max\_digit += 10

temp = digits[:]

digits = temp + [0] \* 10

digits.append(digit)

index += 1

number //= 10

largest = second\_largest = -1

for digit in digits:

if digit > largest:

second\_largest, largest = largest, digit

elif digit > second\_largest and digit != largest:

second\_largest = digit

print(f"Largest digit: {largest}")

print(f"Second Largest digit: {second\_largest}")

4)#include <stdio.h>

#include <stdlib.h>

void findLargestAndSecondLargest(int \*digits, int size, int \*largest, int \*secondLargest) {

\*largest = \*secondLargest = -1;

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

if (digits[i] > \*largest) {

\*secondLargest = \*largest;

\*largest = digits[i];

} else if (digits[i] > \*secondLargest && digits[i] != \*largest) {

\*secondLargest = digits[i];

}

}

}

int main() {

int maxDigit = 10; // Initial size of the array

int \*digits = (int \*)malloc(maxDigit \* sizeof(int));

if (digits == NULL) {

printf("Memory allocation failed!\n");

return 1;

}

int num, index = 0;

printf("Enter a number: ");

scanf("%d", &num);

while (num > 0) {

if (index == maxDigit) {

maxDigit += 10; // Increase the array size by 10

int \*temp = (int \*)realloc(digits, maxDigit \* sizeof(int));

if (temp == NULL) {

printf("Memory reallocation failed!\n");

free(digits);

return 1;

}

digits = temp;

}

digits[index++] = num % 10;

num /= 10;

}

int largest, secondLargest;

findLargestAndSecondLargest(digits, index, &largest, &secondLargest);

printf("Largest digit: %d\n", largest);

printf("Second largest digit: %d\n", secondLargest);

free(digits);

return 0;

}

5)def reverse\_number():

"""Program to take a number as input and reverse it."""

number = int(input("Enter a number: "))

digits = []

while number > 0:

digits.append(number % 10)

number //= 10

print("Reversed Number:", "".join(map(str, digits)))

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

print("1. Employee Bonus Calculation")

employee\_bonus()

print("\n2. Find Youngest and Tallest Friend")

youngest\_and\_tallest()

print("\n3. Find Largest and Second Largest Digits")

find\_largest\_numbers()

print("\n4. Reverse a Number")

reverse\_number()

6) #include <stdio.h>

#include <stdlib.h>

// Function to determine weight status based on BMI

const char\* getWeightStatus(float bmi) {

if (bmi < 18.5) return "Underweight";

else if (bmi >= 18.5 && bmi < 24.9) return "Normal weight";

else if (bmi >= 25 && bmi < 29.9) return "Overweight";

else return "Obese";

}

int main() {

int n;

printf("Enter the number of persons: ");

scanf("%d", &n);

float \*weight = (float \*)malloc(n \* sizeof(float));

float \*height = (float \*)malloc(n \* sizeof(float));

float \*bmi = (float \*)malloc(n \* sizeof(float));

char \*\*status = (char \*\*)malloc(n \* sizeof(char \*));

if (weight == NULL || height == NULL || bmi == NULL || status == NULL) {

printf("Memory allocation failed!\n");

return 1;

}

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

printf("Enter weight (kg) for person %d: ", i + 1);

scanf("%f", &weight[i]);

printf("Enter height (m) for person %d: ", i + 1);

scanf("%f", &height[i]);

bmi[i] = weight[i] / (height[i] \* height[i]);

status[i] = (char \*)getWeightStatus(bmi[i]);

}

printf("\n%-10s %-10s %-10s %-15s\n", "Height(m)", "Weight(kg)", "BMI", "Status");

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

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

printf("%-10.2f %-10.2f %-10.2f %-15s\n", height[i], weight[i], bmi[i], status[i]);

}

free(weight);

free(height);

free(bmi);

free(status);

return 0;

}

7) #include <stdio.h>

#include <stdlib.h>

// Function to determine weight status based on BMI

const char\* getWeightStatus(double bmi) {

if (bmi < 18.5) return "Underweight";

else if (bmi >= 18.5 && bmi < 24.9) return "Normal weight";

else if (bmi >= 25 && bmi < 29.9) return "Overweight";

else return "Obese";

}

int main() {

int n;

printf("Enter the number of persons: ");

scanf("%d", &n);

double personData[n][3]; // Stores height, weight, and BMI

char \*weightStatus[n];

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

do {

printf("Enter weight (kg) for person %d: ", i + 1);

scanf("%lf", &personData[i][0]);

if (personData[i][0] <= 0) {

printf("Weight must be a positive value. Please enter again.\n");

}

} while (personData[i][0] <= 0);

do {

printf("Enter height (m) for person %d: ", i + 1);

scanf("%lf", &personData[i][1]);

if (personData[i][1] <= 0) {

printf("Height must be a positive value. Please enter again.\n");

}

} while (personData[i][1] <= 0);

personData[i][2] = personData[i][0] / (personData[i][1] \* personData[i][1]);

weightStatus[i] = (char \*)getWeightStatus(personData[i][2]);

}

printf("\n%-10s %-10s %-10s %-15s\n", "Height(m)", "Weight(kg)", "BMI", "Status");

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

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

printf("%-10.2f %-10.2f %-10.2f %-15s\n", personData[i][1], personData[i][0], personData[i][2], weightStatus[i]);

}

return 0;

}

8) #include <stdio.h>

#include <stdlib.h>

// Function to determine grade based on percentage

char getGrade(double percentage) {

if (percentage >= 90) return 'A';

else if (percentage >= 80) return 'B';

else if (percentage >= 70) return 'C';

else if (percentage >= 60) return 'D';

else return 'F';

}

int main() {

int n;

printf("Enter the number of students: ");

scanf("%d", &n);

double studentData[n][4]; // Stores Physics, Chemistry, Maths marks, and Percentage

char grades[n];

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

for (int j = 0; j < 3; j++) {

const char \*subjects[] = {"Physics", "Chemistry", "Maths"};

do {

printf("Enter marks in %s for student %d: ", subjects[j], i + 1);

scanf("%lf", &studentData[i][j]);

if (studentData[i][j] < 0) {

printf("Marks must be a positive value. Please enter again.\n");

}

} while (studentData[i][j] < 0);

}

// Calculate percentage

studentData[i][3] = (studentData[i][0] + studentData[i][1] + studentData[i][2]) / 3.0;

// Determine grade

grades[i] = getGrade(studentData[i][3]);

}

printf("\n%-10s %-10s %-10s %-12s %-10s\n", "Physics", "Chemistry", "Maths", "Percentage", "Grade");

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

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

printf("%-10.2f %-10.2f %-10.2f %-12.2f %-10c\n", studentData[i][0], studentData[i][1], studentData[i][2], studentData[i][3], grades[i]);

}

return 0;

}

9) #include <stdio.h>

#include <stdlib.h>

// Function to determine grade based on percentage

char getGrade(double percentage) {

if (percentage >= 90) return 'A';

else if (percentage >= 80) return 'B';

else if (percentage >= 70) return 'C';

else if (percentage >= 60) return 'D';

else return 'F';

}

int main() {

int n;

printf("Enter the number of students: ");

scanf("%d", &n);

double studentMarks[n][3]; // Stores Physics, Chemistry, Maths marks

double percentages[n];

char grades[n];

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

for (int j = 0; j < 3; j++) {

const char \*subjects[] = {"Physics", "Chemistry", "Maths"};

do {

printf("Enter marks in %s for student %d: ", subjects[j], i + 1);

scanf("%lf", &studentMarks[i][j]);

if (studentMarks[i][j] < 0) {

printf("Marks must be a positive value. Please enter again.\n");

}

} while (studentMarks[i][j] < 0);

}

// Calculate percentage

percentages[i] = (studentMarks[i][0] + studentMarks[i][1] + studentMarks[i][2]) / 3.0;

// Determine grade

grades[i] = getGrade(percentages[i]);

}

printf("\n%-10s %-10s %-10s %-12s %-10s\n", "Physics", "Chemistry", "Maths", "Percentage", "Grade");

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

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

printf("%-10.2f %-10.2f %-10.2f %-12.2f %-10c\n", studentMarks[i][0], studentMarks[i][1], studentMarks[i][2], percentages[i], grades[i]);

}

return 0;

}

10) #include <stdio.h>

#include <stdlib.h>

// Function to determine the frequency of digits in a number

void findDigitFrequency(int num) {

int frequency[10] = {0}; // Array to store the frequency of digits (0-9)

int digits[20], count = 0;

// Extract digits and store them in an array

while (num > 0) {

digits[count] = num % 10;

frequency[digits[count]]++;

num /= 10;

count++;

}

// Display the frequency of each digit

printf("\nDigit Frequency:\n");

for (int i = 0; i < 10; i++) {

if (frequency[i] > 0) {

printf("%d: %d\n", i, frequency[i]);

}

}

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (num < 0) {

printf("Please enter a positive number.\n");

return 1;

}

findDigitFrequency(num);

return 0;

}