**SECTION-C(Coding Question) (1x5 marks=5 marks)**

**Q 1. Electricity Bill**

Q1.Write a program in C to calculate and print the electricity bill of a given customer. The customer ID and unit consumed by the user should be taken as input. The program should display the total amount to be paid. The charges are as follows:

Unit Range Charge/Unit

Upto 199 1.20

200 to 400 1.50

401 to 600 1.80

Above 600 2.00

If the bill exceeds Rs. 400, a surcharge of 15% will be applied.

The minimum bill amount is Rs. 100.

Input:

A single integer value representing the number of units consumed.

Output:

A floating-point value representing the total amount, printed to two decimal places.

**Input Explanation**

Input consists of one integer value as number of units consumed by customer

**Output Explanation**

Output consists of float value as due amount with two decimal only.

Test case 1 :

Input:

711

Output:

1635.30

Test case 2 :

Input:

233

Output:

349.50

Test case 3 :

Input:

134

Output:

160.80

Test case 4 :

Input:

468

Output:

968.76

Test case 5 :

Input:

50

Output:

100.00

Test case 6 :

Input:

402

Output:

832.14

**Solution :**

#include <stdio.h>

int main() {

int units;

float bill, surcharge;

const float rate1 = 1.20, rate2 = 1.50, rate3 = 1.80, rate4 = 2.00;

const float surchargeRate = 0.15;

const int minBill = 100;

scanf("%d", &units);

if (units <= 199) {

bill = units \* rate1;

} else if (units <= 400) { // Fixed range to include 200-400

bill = units \* rate2;

} else if (units <= 600) {

bill = units \* rate3;

} else {

bill = units \* rate4;

}

if (bill > 400) {

surcharge = bill \* surchargeRate;

bill += surcharge;

}

if (bill < minBill) {

bill = minBill;

}

printf("%.2f", bill);

return 0;

}

**Q 2 Pre-primary school (vowel or consonant)**

A pre-primary school wants a program to help students determine whether the entered character is a vowel or a consonant. Write a C program that takes a single character as input and determines if it is a vowel or consonant. Ensure the program handles both uppercase and lowercase letters.

Input:

A single character (either uppercase or lowercase). Output:

Print <character> is Vowel if it is a vowel.

Print <character> is Consonant if it is a consonant. Print Invalid Input if the character is not an alphabet.

Sample Input

S

Sample Output

S is Consonant

Input Explanation

Input consists of single “character” value which accept both Uppercase and Lowercase.

Output Explanation

Consists of text format only.

Test case 1 : Input:

9

Output:

Invalid Input

Test case 2 : Input:

A

Output:

A is Vowel

Test case 3 : Input:

R

Output:

R is Consonant

Test case 4 : Input:

P

Output:

P is Consonant

Test case 5 : Input:

U

Output:

U is Vowel Test case 6 :

Input:

8

Output:

Invalid Input

**Solution 2:**

#include <stdio.h>

int main() {

char c;

scanf("%c", &c);

if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||

c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U') {

printf("%c is Vowel\n", c);

} else if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {

printf("%c is Consonant\n", c);

} else {

printf("Invalid Input\n");

}

return 0;

}

**Q3. Decoding the lock**

At "YoursPay" Bank, a digital lock is encoded with a five-digit numeric code. To decode the lock: Count the number of even and odd digits in the encoded number.

If the count of even digits is greater, the decoding code is the sum of the even digits.

If the count of odd digits is greater, the decoding code is the sum of the odd digits.

Write a program to help the bank decode the lock.

Input:

A five-digit integer value (n).

Output:

An integer representing the decoded value.

**Input Explanation**

Input consists of five digits integer value

For input 40722:

Even digits: 4, 0, 2, 2 → Count = 4, Sum = 8. Odd digits: 7 → Count = 1, Sum = 7.

The input number can be of any length

**Output Explanation**

Output consists of integer value

Since even count > odd count, output is 8.

Example:

Input:

10

Output:

1

Test case 1 :

Input:

40722

Output:

8

Test case 2 :

Input:

13621

Output:

5

Test case 3 :

Input:

16328

Output:

16

Test case 4 :

Input:

37003

Output:

13

Test case 5 :

Input:

24574

Output:

10

Test case 6 :

Input:

13645

Output:

9

**Solution 3:**

#include <stdio.h>

int main() {

int n, sum\_even = 0, sum\_odd = 0, count\_even = 0, count\_odd = 0, digit;

scanf("%d", &n);

while (n != 0) {

digit = n % 10; // Extract the last digit

if (digit % 2 == 0) {

sum\_even += digit;

count\_even++;

} else {

sum\_odd += digit;

count\_odd++;

}

n /= 10; // Remove the last digit

}

if (count\_even > count\_odd) {

printf("%d\n", sum\_even);

} else {

printf("%d\n", sum\_odd);

}

return 0;

}

**Q4. California Fire Report**

**Problem Statement:**

Firefighters in California log fire reports daily, including the number of active fires. Write a program to count how many days had more than X fires. The program should accept:

1. The number of days, N.
2. The threshold value, X.
3. The number of fires for each day.

Print the count of days exceeding the threshold. Use a while loop for the implementation.

**Sample Input and Output:**

**Input:**

5 // NO OF DAYS

50 // THRESHOLD VALUE

40 //NO OF FIRES EACH DAY

60

30

70

20

**Output:**

2

**Input:**

**0**

**Output:**

**Invalid Input**

Test case 1 : Input:

3

25

10

20

30

Output:

1

Test case 2 : Input:

5

50

50

60

30

22

70

Output:

2

Test case 3 :

Input:

3

25

10

20

30

Output:

1

Test case 4 :

Input:

0

Output:

Invalid Input

Test case 5 :

Input:

4

15

10

20

25

15

Output:

2

Test case 6 :

Input:

2

10

5

8

Output:

0

**Solution 4:**

#include <stdio.h>

int main() {

int n, threshold, count = 0, i = 1, fires;

scanf("%d", &n);

if (n <= 0) {

printf("Invalid Input\n");

} else {

scanf("%d", &threshold);

while (i <= n) {

scanf("%d", &fires);

if (fires > threshold) {

count++;

}

i++;

}

printf("%d", count);

}

return 0;

}

**Q 5.**  Once upon a time, a curious student Manu sat in front of their computer, eager to learn about leap years. "Is this year a leap year?" they wondered aloud. The wise computer, always ready with an answer, replied with a simple rule. "A year is a leap year if it’s divisible by 4, but not by 100, unless it’s also divisible by 400." Intrigued, the student thought deeply about the rule, repeating it to themselves. They realized how clever this logic was and felt proud of the newfound knowledge. From that day on, they could easily identify a leap year with confidence.

**INPUT:**

The program accepts a single integer input from the user, which represents the year to be checked.

**OUTPUT:**

The program outputs a message indicating whether the given year is a leap year or not.

Example :

Input: 2050

Output: 2050 is NOT a Leap Year.

**Solution:**

#include <stdio.h>

int main() {

int year;

scanf("%d", &year);

// Check if the year is a leap year

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

printf("%d is a Leap Year.\n", year);

} else {

printf("%d is NOT a Leap Year.\n", year);

}

return 0;

}

**Q 6 Question paper Set**

In an online exam, test paper sets are categorized using letters A-Z. Each student is assigned a 6-digit roll number. The exam set code is determined as follows:

Calculate the sum of the digits in the roll number. If the sum exceeds 26, repeatedly calculate the digit sum until it falls in the range [1, 26]. Map the final sum to the corresponding letter (e.g., 1 → A, 2 → B, ..., 26 → Z).

Write a program to calculate and display the exam set code based on the roll number.

Input:

A single integer representing the roll number.

Output:

A single uppercase letter corresponding to the exam set code.

Sample Input

6442

Sample Output

P

**Input Explanation**

Input consists one integer value as Roll Number.

**Output Explanation**

The sum of the digits of the Roll Number is 6 + 4 + 4 + 2 = 16. The letter that corresponds to 16 is 'P'. Hence the output is P.

**Solution:**

#include <stdio.h>

int main() {

int rollNumber, sum = 0;

scanf("%d", &rollNumber);

// Calculate the sum of digits

while (rollNumber != 0) {

sum += rollNumber % 10;

rollNumber /= 10;

}

// Map the numeric sum to the corresponding alphabet

printf("%c\n", 'A' + (sum - 1));

return 0;

}

**Q7. Movie Theatre Ticket Counter Analysis**

**Statement:**

A movie theatre has a ticket counter where tickets for different movies are sold. The counter records the total earnings for N transactions. Write a program to calculate the total earnings from all transactions until the counter reaches a negative transaction, which indicates a refund error. Use a while loop to process each transaction.

· If a negative transaction is encountered, terminate the input and print **Error Detected** along with the earnings before the error.

· If no errors occur, print the total earnings.

· If there is no transactions for the day print **Invalid Input.**

**Sample Input and Output:**

**Input:**

5

500

1000

200

-100

**Output:**

Error Detected

1700

**Explanation:**

In the first example, the program calculates the earnings until the refund error is encountered. The sum of 500+1000+200=1700 and error is also detected.

**Solution:**

#include <stdio.h>

int main() {

int n, transaction, totalEarnings = 0, i = 1;

scanf("%d", &n);

if (n <= 0) {

printf("Invalid Input\n");

return 0;

}

while (i <= n) {

scanf("%d", &transaction);

if (transaction < 0) {

printf("Error Detected\n%d", totalEarnings);

return 0;

}

totalEarnings += transaction;

i++;

}

printf("%d", totalEarnings);

return 0;

}

**Q8. Armstrong Number**

One day, a curious student Manu encountered a number that seemed special, and they wondered if it was an Armstrong number. They decided to ask their magical program. The program explained, "An Armstrong number is one where the sum of its digits raised to the power of the number of digits equals the number itself." Excited, the student typed in the number, and the program started calculating. If the sum of the digits, each raised to the power of how many digits there were, matched the number, it would declare, "This is an Armstrong number!" If not, it would say, "This is not an Armstrong number."

**INPUT:**

The program accepts a single integer as input, which represents the number to be checked.

**OUTPUT:**

The program outputs a message indicating whether the number is an Armstrong number or not.

**Solution:**

#include <stdio.h>

#include <math.h> // For pow() function

int main() {

int number, originalNumber, remainder, n = 0;

double result = 0.0;

// Accept input from the user

// printf("Enter a number: ");

scanf("%d", &number);

originalNumber = number;

// Count the number of digits

while (originalNumber != 0) {

originalNumber /= 10;

++n;

}

originalNumber = number;

// Compute the sum of powers of digits

while (originalNumber != 0) {

remainder = originalNumber % 10;

result += pow(remainder, n);

originalNumber /= 10;

}

// Check if the number is an Armstrong number

if ((int)result == number) {

printf("%d is an Armstrong number.\n", number);

} else {

printf("%d is NOT an Armstrong number.\n", number);

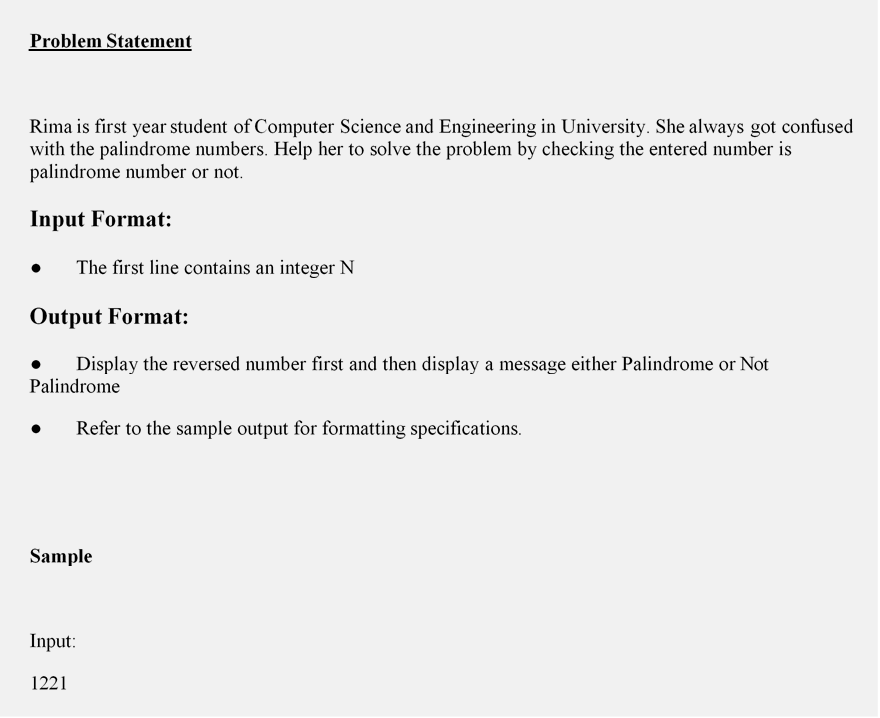
}

return 0;

}

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q1 **Palindrome**

****

Output:

1221 Palindrome

Test case 1 : Input:

1212

Output:

2121 Not Palindrome

Test case 2 : Input:

1111

Output:

1111 Palindrome

Test case 3 : Input:

5252

Output:

2525 Not Palindrome

Test case 4 : Input:

8998

Output:

8998 Palindrome

Test case 5 : Input:

9182

Output:

2819 Not Palindrome

Test case 6 : Input:

7678

Output:

8767 Not Palindrome

**Solution:**

#include <stdio.h>

int main()

{

int n, rev = 0, rem, num;

scanf("%d", &n);

num = n;

while (n != 0)

{

rem = n % 10;

rev = rev \* 10 + rem;

n /= 10;

}

if (num == rev)

printf("%d Palindrome", rev);

else

printf("%d Not Palindrome", rev);

return 0;

}

Q2. Employee Bonus Calculation

Problem Statement:

A company provides bonuses to employees based on their performance rating:

● Rating 5: "Excellent" bonus (20% of the salary).

● Rating 4: "Good" bonus (15% of the salary).

● Rating 3: "Average" bonus (10% of the salary).

● Rating 2 or below: "No Bonus."

Write a program that takes an employee's performance rating (integer) and salary (float) as input and calculates the bonus amount. If the rating value is invalid, print "Invalid Rating."

Sample Input and Output:

Input: 5, 50000.0

Output: 10000.0

Test case 1 :

Input:

6, 45000.0

Output:

Invalid Rating

Test case 2 : Input:

5

50000.0

Output:

10000.0

Test case 3 : Input:

4

40000.0

Output:

6000.0

Test case 4 :

Input:

3

30000.0

Output:

3000.0

Test case 5 : Input:

2

25000.0

Output:

0.0

Test case 6 : Input:

6

45000.0

Output:

Invalid Rating

**Solution.**

The input consists of a performance rating and salary. The program calculates the bonus based on the rating.

#include <stdio.h>

int main() {

int rating;

float salary, bonus = 0;

//printf("Enter performance rating and salary: ");

scanf("%d", &rating);

scanf("%f",&salary);

if (rating < 1 || rating > 5) {

printf("Invalid Rating\n");

} else {

if (rating == 5) bonus = salary \* 0.2;

else if (rating == 4) bonus = salary \* 0.15;

else if (rating == 3) bonus = salary \* 0.1;

printf("%.1f\n", bonus);

}

return 0;

}

Q3 **Triangle Type Identifier**

You are developing a C program that determines the type of triangle based on its side lengths provided by the user. The triangle types are defined as follows:

**Equilateral triangle** if all three sides are equal.

**Isosceles triangle** if exactly two sides are equal.

**Scalene triangle** if all three sides are different.

**Invalid triangle** if the sum of any two sides is less than or equal to the third side.

Additionally, your program should handle negative inputs. The program breaks out of the loop once it successfully identifies the type of triangle or determines that it is invalid.

Write a program that reads three integers representing the lengths of the sides of a triangle and outputs its type.

**Input Format:**

Three integers representing the lengths of the sides of a triangle.

**Output Format:**

Print "Equilateral", "Isosceles", "Scalene", or "Invalid". However if there are negative integers entered in input then print “Lengths must be positive integers”.

**Explanation:**

The program uses nested if statements to check for validity and determine the type of triangle.

**Example:**

**Input:**

-3 -4 -5 //Lengths of three sides

**Output:**

Lengths must be positive integers.

**Solution:**

#include <stdio.h>

int main() {

int a, b, c;

while (1) {

scanf("%d %d %d", &a, &b, &c);

if (a <= 0 || b <= 0 || c <= 0) {

printf("Lengths must be positive integers.\n");

break;

}

if (a + b <= c || a + c <= b || b + c <= a) {

printf("Invalid\n");

break;

} else if (a == b && b == c) {

printf("Equilateral\n");

break;

} else if (a == b || b == c || a == c) {

printf("Isosceles\n");

break;

} else {

printf("Scalene\n");

break;

}

}

return 0;

}

Q4. **Timetable Conflict Checker**

A university is planning its timetable and wants to ensure there are no conflicts between classes. Write a program that checks if two classes overlap based on their start and end times. The program takes two pairs of integers (start1, end1) and (start2, end2) as input, where each pair represents the start and end times of a class in 24-hour format.

●       If the time intervals overlap, print **"Conflict"**.

●       Otherwise, print **"No Conflict"**.

Ensure the inputs are valid (e.g., start time is less than end time). If invalid, print **"Invalid Input"**.

**Sample Input and Output:**

**Input:**

10

12

11

13

**Output:**

Conflict

**Explanation:**

In the example, Class 1 runs from 10 to 12, and Class 2 runs from 11 to 13. Since their timings overlap (11 ≤ 12), the output is "Conflict."

**Solution:**

#include <stdio.h>

int main() {

int start1, end1, start2, end2;

scanf("%d %d", &start1, &end1);

scanf("%d %d", &start2, &end2);

if (start1 >= end1 || start2 >= end2) {

printf("Invalid Input\n");

} else if ((start1 < end2 && start2 < end1)) {

printf("Conflict\n");

} else {

printf("No Conflict\n");

}

return 0;

}

Q5. **Palindrome numbers in the given range.**

Abhi is curious about learning the palindrome concept. Reverse is easy but he wants to learn it finding within a range. Help Abhi to write a program in C to print the Palindrome numbers in the given range.

**Input Format:**

The first line contains an integer N to provide the lower range

The 2nd line contains an integer N to provide the upper range

**Output Format:**

Single line print of each palindromes number from the given range

**Sample:**

Input:

11

121

Output:

11 22 33 44 55 66 77 88 99 101 111 121

**Solution:**

 #include<stdio.h>

int main()

{

int num, rem, reverse\_num, temp, start, end, count;

scanf("%d",&start);

scanf("%d",&end);

for(num=start;num<=end;num++)

{

temp=num;

reverse\_num=0;

while(temp)

{

rem=temp%10;

temp=temp/10;

reverse\_num=reverse\_num\*10+rem;

}

if(num==reverse\_num)

{

printf("%d ",num);

count++;

}

}

return 0;

}