CAT-1

NAME: JITHENDRA BOODATI

REG NO:22BCE1947

1 .The temperature in Chennai exists somewhere between 280C to 330C in a particular week. The temperature in Delhi is 8 0C lesser than in Chennai. Likewise, the temperature in Haveri is 5 0C more than that of Chennai. Write a C program to find the temperature of Gangtok for a particular week, which is the temperature difference between Delhi and Haveri. Get the temperature of Chennai (0C) as input for the week of 7 days and the temperature of Gangtok (0C) as output for the week of 7 days. Implement the program using arrays and validate the rules

CODE:

```
#include <stdio.h>


int main() {
   float chennai[7], delhi[7], haveri[7], gangtok[7];
float chennai_temp, gangtok_temp;
   printf("Enter temperature for Chennai for 7 days (in degrees
Celsius):\n");
 for (int i = 0; i < 7; i++)
{
 scanf("%f", &chennai[i]);
   if (chennai[i] < 28 || chennai[i] > 33)
{
```

```
        printf("Temperature should be between 28C and 33C. Please enter
again.\n");
i--;
      }
   }
   for (int i = 0; i < 7; i++) {
      delhi[i] = chennai[i] - 8;
haveri[i] = chennai[i] + 5;
  }
   for (int i = 0; i < 7; i++) {
gangtok[i] = haveri[i] - delhi[i];
   }
   printf("Temperature for Gangtok for 7 days (in degrees
Celsius):\n");    for (int i = 0; i < 7; i++) {       printf("%.2f\n",
gangtok[i]);
   }
   return 0;
}
```

2. Samantha is an avid collector of lucky numbers. She believes that each number has its own unique energy and can bring good luck if used correctly. One day, she came across a new number that had a mysterious aura and she could not resist finding out more about it. She heard that the number may fall within the digit combinations, so she decided to sum up all the four-digit even numbers and then keep adding the digits of the summation until a single digit is found to unlock its secrets. Later, she must check again whether the single digit is odd or even. If odd, then you must say "Odd Found"

otherwise you must return “Even found”. Write a C program to help her in finding the mysterious number.

CODE:

```
#include <stdio.h>


int main() {
   int sum = 0, digit_sum = 0, num;
      for (int i = 1000; i <= 9998; i += 2) {
sum += i;
   }
   while (sum > 0 || digit_sum > 9) {
if (sum == 0)
{
 sum = digit_sum;
        digit_sum = 0;
       }
       digit_sum += sum % 10;
sum /= 10;
   }
   if (digit_sum % 2 == 0) {
printf("Even found");
   } else {
      printf("Odd found");
```

```
    }
    return 0;
}
```

3 .A digital locker in the bank is protected with a security mechanism. To open the locker a password of 9 characters is required. The input characters should be accepted as 3x3 matrix and two diagonal characters of the matrix are concatenated (refer to the example given below) and compared with the password already stored in a character array for authentication. Write a C program to implement this logic for password verification.

Input to open the device a b c d e f g h i

Concatenation of Diagonal characters: aeiceg CODE:

```
#include <stdio.h>
#include <string.h>

int main() {
    char password[10] = "aeiceg";
char input[3][3];
printf("Enter the input matrix (3x3):\n");
 for (int i = 0; i < 3; i++)
{
 for (int j = 0; j < 3; j++)
{
 scanf(" %c", &input[i][j]);
       }
```

```
    }
    char diagonal[7];
diagonal[0] = input[0][0];
diagonal[1] = input[1][1];
diagonal[2] = input[2][2];
diagonal[3] = input[0][2];
diagonal[4] = input[1][1];
diagonal[5] = input[2][0];
diagonal[6] = '\0';
    if (strcmp(diagonal, password) == 0) {
printf("Password verified. Locker opened.\n");
    } else {
        printf("Incorrect password. Locker remains locked.\n");
    }
    return 0;
}
```

4 .An international school of class 7 with a class strength of 25, decided to assign additional marks for their students in Maths subject, to increase their class average. The additional marks were given to each student based on their month of birth. That is she was to give that number (month of birth) as the booster marks. The teacher wants to find the class average for the original marks as well as for the revised marks. She 10 marks wants to decide whether to implement this revision in marks or not based on the significant improvement in the class average. Write a C program to help the teacher get the class average for the original marks as well as the revised marks. She wants to know whether to implement this revision or not. This is decided based on the condition that the revision should bring a significant increase in the class average of 5 marks. Else, she is not planning to implement this

revision in the marks strategy. Write the program to display this decision of "Can implement – Significant increase in class average" or "Need not implement – No significant increase in class average". Keep every operation in this program separate. Get the students' original marks and the month of their birth as input.

CODE:

```
#include <stdio.h>


int main()

{

int

marks[25];

 int revised_marks[25];

int birth_month[25];

int sum = 0;

 int revised_sum = 0;

   printf("Enter the original marks and birth months of the 25 students:\n");

for (int i = 0; i < 25; i++) {

      scanf("%d %d", &marks[i], &birth_month[i]);

   }

      for (int i = 0; i < 25; i++) {

      revised_marks[i] = marks[i] + birth_month[i];

sum += marks[i];

      revised_sum += revised_marks[i];

   }
```

```
    float avg = (float) sum / 25.0;
    float revised_avg = (float) revised_sum / 25.0;
if (revised_avg - avg >= 5.0) {
        printf("Can implement - Significant increase in class average\n");
    } else {
        printf("Need not implement - No significant increase in class
average\n");
    }
    return 0;
}
```

5 .Paul is provided with a number "x" whose scope will remain throughout the program. Using the concept of recursion, help Paul to write a "C program" for finding the value of (x)n where n should be less than or equal to 5 CODE:

```
#include <stdio.h>


int power(int x, int n);
int main() {     int x, n,
result;
    printf("Enter the value of x: ");
scanf("%d", &x);
    printf("Enter the value of n: ");
scanf("%d", &n);

    if (n <= 5)
{
```

```
result = power(x, n);
printf("%d^%d = %d\n", x, n, result);
   } else {
      printf("Error: n must be less than or equal to 5.\n");
   }
   return 0;
}
int power(int x, int n)
 {
if (n == 0)
 {
return
1;    } else
{
return x * power(x, n-1);
   }
}
```

** **IPS**:

Get the three angles of a triangle as input.

Find The count of the type of the triangle.

Continue the process for 5 times.

If the sum of the three angles is greater than 180 then prompt for correct values. (the sum of all internal angles of a triangle is always equal to 180°). Keep the count of the wrong entries also.

Acute Angled Triangle (all three angles less than 90°)

Right-Angled Triangle (one angle that measures exactly 90°)

Obtuse Angled Triangle (one angle that measures more than 90°)

Sample i/p:

60

70

50

40

50

90

40

40

100

30

30

120

90

60

30

Sample o/p:

Acute Angled Triangle: 1

Right Angled Triangle: 2

Obtuse Angled Triangle: 2

Wrong Entries: 0

Second Sample i/p

60

70

50

40

50

90

40

40

100

30

30

120

90

90

30

Wrong Entry try again

90

30

60

Sample o/p:

Wrong Entry try again

Acute Angled Triangle: 1

Right Angled Triangle: 2

Obtuse Angled Triangle: 2

Wrong Entries: 1

CODE:

```
#include <stdio.h>

int main() {
   int i, sum, angle1, angle2, angle3;
   int acute_count = 0, right_count = 0, obtuse_count = 0, wrong_count = 0;
   for (i = 1; i <= 5; i++) {
```

```
        printf("Enter the three angles of triangle %d: ", i);
scanf("%d%d%d", &angle1, &angle2, &angle3);
        sum = angle1 + angle2 + angle3;
        if (sum <= 180) {
            if (angle1 < 90 && angle2 < 90 && angle3 < 90) {
acute_count++;
            } else if (angle1 == 90 || angle2 == 90 || angle3 == 90) {
right_count++;

            } else {
                obtuse_count++;
            }
        } else {
            printf("Wrong Entry try again\n");
i--;
            wrong_count++;
        }
    }
    printf("\nAcute Angled Triangle: %d\n", acute_count);
printf("Right Angled Triangle: %d\n", right_count);     printf("Obtuse
Angled Triangle: %d\n", obtuse_count);
    printf("Wrong Entries: %d\n", wrong_count);

    return 0;
}
```