1.Write a C program to compute the roots of a quadratic equation

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

int main() {
   float a, b, c;
   float discriminant, realPart, imagPart, root1, root2;

// Input
printf("Enter coefficients a, b and c: ");
scanf("%f %f %f", &a, &b, &c);
// Calculate discriminant
discriminant = b * b - 4 * a * c;
if (discriminant > 0) {
// Real and distinct roots
root1 = (-b + sqrt(discriminant)) / (2 * a);
root2 = (-b - sqrt(discriminant)) / (2 * a);
printf("Roots are real and distinct.\n");
printf("Root1 = %.2f, Root2 = %.2f\n", root1, root2);
}
else if (discriminant == 0) {
// Real and equal roots
root1 = root2 = -b / (2 * a);
printf("Roots are real and equal.\n");
printf("Root1 = Root2 = %.2f\n", root1);
}
else {
// Complex roots
realPart = -b / (2 * a);
imagPart = sqrt(-discriminant) / (2 * a);
printf("Roots are complex.\n");
printf("Root1 = %.2f + %.2fi\n", realPart, imagPart);
printf("Root2 = %.2f - %.2fi\n", realPart, imagPart);
}
return 0;
}
```

2.Write a C program to develop a billing software module for an electricity supply company

```
#include <stdio.h>
int main() {
char name[50];
int units;
float bill, surcharge = 0;
// Input
printf("Enter customer name (no spaces): ");
scanf("%s", name);
printf("Enter units consumed: ");
scanf("%d", &units);
// Calculate bill
if (units <= 200) {
bill = units * 0.80;
} else if (units <= 300) {
bill = (200 * 0.80) + (units - 200) * 0.90;
} else {
bill = (200 * 0.80) + (100 * 0.90) + (units - 300) * 1.00;
}
// Add surcharge if bill > 400
if (bill > 400) {
surcharge = 0.15 * bill;
}
// Add minimum meter charge
bill = bill + surcharge + 100;
// Output
printf("\nCustomer: %s\n", name);
printf("Total Bill = Rs. %.2f\n", bill);
return 0;
}
```

3.The bank's security system needs to verify cheque numbers entered by customers.

```
#include <stdio.h>
int main() {
int n, temp, rev = 0, digit;
printf("Enter a number: ");
scanf("%d", &n);
temp = n;
while (n > 0) {
digit = n % 10;
rev = rev * 10 + digit;
n = n / 10;
}
if (rev == temp) {
printf("It is a palindrome\n");
} else {
printf("It is not a palindrome\n");
}
return 0;
}
```

4. Factorial of a Number

```
#include <stdio.h>
// Recursive function to calculate factorial
long long factorial(int n) {
if (n == 0 || n == 1)
return 1;
else
return n * factorial(n - 1);
}
int main() {
int n;
printf("Enter a number: ");
scanf("%d", &n);
if (n < 0) {
printf("Factorial is not defined for negative numbers.\n");
} else {
printf("Factorial of %d = %lld\n", n, factorial(n));
}
return 0;
}
```

5. Write a C program to search for a given element using binary search technique.

```
#include <stdio.h>

int main() {
    int n, key, low, high, mid, found = -1;

    printf("Enter number of books: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d sorted book IDs: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter book ID to search: ");
    scanf("%d", &key);

    low = 0;
    high = n - 1;

    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] == key) {
            found = mid;
            break;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    if (found != -1)
        printf("Book found at position %d\n", found);
    else
        printf("Book not found\n");

    return 0;
}
```

6. Write a C program to sort the given array element in ascending order.

```
 #include <stdio.h>
int main() {
int n, i, j, temp;
printf("Enter number of transactions: ");
scanf("%d", &n);
int arr[n];
printf("Enter %d transaction amounts: ", n);
for (i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
// Bubble Sort
for (i = 0; i < n - 1; i++) {
for (j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;
}
}
}
printf("Sorted transaction amounts: ");
for (i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
printf("\n");
return 0;
}
```

7. String Comparision

```
#include <stdio.h>
// Function to find string length
int strLength(char str[]) {
int len = 0;
while (str[len] != '\0')
len++;
return len;
}
// Function to compare two strings
int strCompare(char str1[], char str2[]) {
int i = 0;
while (str1[i] != '\0' && str2[i] != '\0') {
if (str1[i] != str2[i])
return 0; // not equal
i++;
}
return (str1[i] == '\0' && str2[i] == '\0');
}

// Function to concatenate two strings
void strConcat(char str1[], char str2[], char result[]) {
    int i = 0, j = 0;
    while (str1[i] != '\0') {
        result[i] = str1[i];
        i++;
    }
    while (str2[j] != '\0') {
        result[i] = str2[j];
        i++;
        j++;
    }
    result[i] = '\0';
}

int main() {
    int choice;
    char str1[100], str2[100], result[200];

    printf("Enter your choice (1-Compare, 2-Concatenate, 3-Length): ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter first string: ");
        scanf("%s", str1);
        printf("Enter second string: ");
        scanf("%s", str2);
```

```
        if (strCompare(str1, str2))
            printf("Strings are equal\n");
        else
            printf("Strings are not equal\n");
    }
    else if (choice == 2) {
        printf("Enter first string: ");
        scanf("%s", str1);
        printf("Enter second string: ");
        scanf("%s", str2);
        strConcat(str1, str2, result);
        printf("Concatenated string: %s\n", result);
    }
    else if (choice == 3) {
        printf("Enter a string: ");
        scanf("%s", str1);
        printf("Length = %d\n", strLength(str1));
    }
    else {
        printf("Invalid choice\n");
    }

    return 0;
}
```

8. Multiplication of two matrices.

```
#include <stdio.h>
int main() {
int m1, n1, m2, n2, i, j, k;
// Input dimensions and first matrix
printf("Enter dimensions of first matrix (m1 n1): ");
scanf("%d %d", &m1, &n1);
int A[m1][n1];
printf("Enter elements of first matrix:\n");
for (i = 0; i < m1; i++) {
for (j = 0; j < n1; j++) {
scanf("%d", &A[i][j]);
}

    }

    // Input dimensions and second matrix
    printf("Enter dimensions of second matrix (m2 n2): ");
    scanf("%d %d", &m2, &n2);

    int B[m2][n2];
    printf("Enter elements of second matrix:\n");
```

```
    for (i = 0; i < m2; i++) {
        for (j = 0; j < n2; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    // Check multiplication possibility
    if (n1 != m2) {
        printf("Matrix multiplication not possible\n");
        return 0;
    }

    int C[m1][n2];

    // Initialize result matrix
    for (i = 0; i < m1; i++) {
        for (j = 0; j < n2; j++) {
            C[i][j] = 0;
        }
    }

    // Perform multiplication
    for (i = 0; i < m1; i++) {
        for (j = 0; j < n2; j++) {
            for (k = 0; k < n1; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    // Display result
    printf("Resultant matrix:\n");
    for (i = 0; i < m1; i++) {
        for (j = 0; j < n2; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

9. C program to implement structures.

```
#include <stdio.h>
#include <string.h>
struct Player {
int id;
char name[50];
int score;
char skill[20];
};
int main() {
int N, i, j;
printf("Enter number of players: ");
scanf("%d", &N);
struct Player players[N];
int total = 0;


    // Input player details
    for (i = 0; i < N; i++) {
        printf("Enter ID, Name (use _ for spaces), and Score: ");
        scanf("%d %s %d", &players[i].id, players[i].name, &players[i].score);
        total += players[i].score;

        // Assign skill level
        if (players[i].score >= 80)
            strcpy(players[i].skill, "Elite");
        else if (players[i].score >= 60)
            strcpy(players[i].skill, "Advanced");
        else if (players[i].score >= 40)
            strcpy(players[i].skill, "Intermediate");
        else
            strcpy(players[i].skill, "Beginner");
    }

    float avg = (float) total / N;
    printf("\nOverall Average Score = %.2f\n", avg);

    // Count categories
    int elite = 0, adv = 0, inter = 0, beg = 0;
    for (i = 0; i < N; i++) {
        if (strcmp(players[i].skill, "Elite") == 0) elite++;
        else if (strcmp(players[i].skill, "Advanced") == 0) adv++;
        else if (strcmp(players[i].skill, "Intermediate") == 0) inter++;
        else beg++;
    }

    printf("Elite: %d, Advanced: %d, Intermediate: %d, Beginner: %d\n",
```

```
        elite, adv, inter, beg);

    // Display all players
    printf("\nPlayer Details:\n");
    for (i = 0; i < N; i++) {
        printf("ID: %d, Name: %s, Score: %d, Skill: %s\n",
            players[i].id, players[i].name, players[i].score, players[i].skill);
    }

    // Sort players by score (descending)
    struct Player temp;
    for (i = 0; i < N - 1; i++) {
        for (j = i + 1; j < N; j++) {
            if (players[j].score > players[i].score) {
                temp = players[i];
                players[i] = players[j];
                players[j] = temp;
            }
        }
    }

// Display top 3 performers
printf("\nTop Performers:\n");
for (i = 0; i < N && i < 3; i++) {
printf("%s with Score: %d\n", players[i].name, players[i].score);
}
return 0;
}
```

10. C program using pointer (Players)

```
#include <stdio.h>
#include <math.h>
int main() {
int n, i;
double sum = 0, mean, variance = 0, stdDev;
printf("Enter number of parts: ");
scanf("%d", &n);
double arr[n];
printf("Enter %d part lengths: ", n);
for (i = 0; i < n; i++) {
scanf("%lf", &arr[i]);
}
// Using pointer for sum
double *ptr = arr;
for (i = 0; i < n; i++) {
sum += *(ptr + i);
}
mean = sum / n;
// Calculate variance
for (i = 0; i < n; i++) {
variance += pow(*(ptr + i) - mean, 2);
}
variance /= n;
stdDev = sqrt(variance);
printf("Sum = %.2f\n", sum);
printf("Mean = %.2f\n", mean);
printf("Standard Deviation = %.2f\n", stdDev);
return 0;
}
```

11. C Program: Advanced Income Tax Computation

```
#include <stdio.h>

int main() {
    float income, investments, taxable_income, tax = 0;
    int age;

    // Input
    printf("Enter annual income: ");
    scanf("%f", &income);

    printf("Enter age: ");
    scanf("%d", &age);

    printf("Enter total investments: ");
    scanf("%f", &investments);

    taxable_income = income;

    // Senior citizen exemption
    if (age >= 60) {
        taxable_income = taxable_income - 50000;
    }

    // Investment deduction
    if (investments > 150000) {
        taxable_income = taxable_income - 75000;
    }

    // Tax slab calculation
    if (taxable_income <= 250000) {
        tax = 0;
    }
    else if (taxable_income <= 500000) {
        tax = taxable_income * 0.05;
    }
    else if (taxable_income <= 1000000) {
        tax = taxable_income * 0.20;
    }
    else {
        tax = taxable_income * 0.30;
    }

    // Minimum tax condition
    if (tax > 0 && tax < 1000) {
        tax = 1000;
    }
```

```
    // Output
    printf("\nTaxable Income: %.2f", taxable_income);
    printf("\nFinal Tax Payable: %.2f\n", tax);

    return 0;
}
```

12. ATM Cash Withdrawal Processing System

```
#include <stdio.h>

/* Function to check PIN attempts */
int checkPinAttempts(int attempts) {
    if (attempts > 3)
        return 0;   // Card blocked
    return 1;       // Card active
}

/* Function to check minimum balance */
int checkMinimumBalance(float balance, int accountType, float withdrawAmount) {
    float minBalance;

    // accountType: 1 = Savings, 2 = Current
    if (accountType == 1)
        minBalance = 1000;
    else
        minBalance = 5000;

    if ((balance - withdrawAmount) < minBalance)
        return 0;   // Insufficient balance
    return 1;
}

/* Function to validate withdrawal amount */
int validateAmount(float amount) {
    if ((int)amount % 100 != 0)
        return 0;   // Not multiple of 100
    if (amount > 25000)
        return 0;   // Exceeds daily limit
    return 1;
}

int main() {
    float balance, withdrawAmount;
    int pinAttempts, accountType;

    // Input
```

```
    printf("Enter account balance: ");
    scanf("%f", &balance);

    printf("Enter withdrawal amount: ");
    scanf("%f", &withdrawAmount);

    printf("Enter number of wrong PIN attempts: ");
    scanf("%d", &pinAttempts);

    printf("Account type (1-Savings, 2-Current): ");
    scanf("%d", &accountType);

    // Rule 1: PIN attempts check
    if (checkPinAttempts(pinAttempts) == 0) {
        printf("\nCARD BLOCKED: Too many incorrect PIN attempts.\n");
        return 0;
    }

    // Rule 2: Withdrawal amount validation
    if (validateAmount(withdrawAmount) == 0) {
        printf("\nWITHDRAWAL FAILED: Invalid amount or exceeds daily limit.\n");
        return 0;
    }

    // Rule 3: Minimum balance check
    if (checkMinimumBalance(balance, accountType, withdrawAmount) == 0) {
        printf("\nWITHDRAWAL FAILED: Minimum balance requirement not met.\n");
        return 0;
    }

    // If all checks passed
    balance = balance - withdrawAmount;

    printf("\nWITHDRAWAL SUCCESSFUL");
    printf("\nRemaining Balance: ₹%.2f\n", balance);

    return 0;
}
```

13. Hospital Patient Admission & Billing System

```
#include <stdio.h>

/* Structure to store patient details */
struct Patient {
    int patientID;
    int age;
    int admissionType;   // 1 = General, 2 = ICU
    int daysAdmitted;
    int insurance;       // 1 = Yes, 0 = No
};

int main() {
    struct Patient p;
    float totalBill = 0, discount = 0, insuranceCover = 0, finalBill = 0;
    float perDayCharge;

    // Input patient details
    printf("Enter Patient ID: ");
    scanf("%d", &p.patientID);

    printf("Enter Age: ");
    scanf("%d", &p.age);

    printf("Admission Type (1-General, 2-ICU): ");
    scanf("%d", &p.admissionType);

    printf("Enter Number of Days Admitted: ");
    scanf("%d", &p.daysAdmitted);

    printf("Insurance Available? (1-Yes, 0-No): ");
    scanf("%d", &p.insurance);

    // Determine per-day charge
    if (p.admissionType == 2)
        perDayCharge = 5000;   // ICU
    else
        perDayCharge = 2000;   // General

    // Calculate total bill
    totalBill = perDayCharge * p.daysAdmitted;

    // Senior citizen discount
    if (p.age > 60) {
        discount = totalBill * 0.10;   // 10% discount
    }
```

```
    // Insurance coverage
    if (p.insurance == 1) {
        insuranceCover = totalBill * 0.70;  // 70% coverage
    }

    // Final bill calculation
    finalBill = totalBill - discount - insuranceCover;

    // Output
    printf("\n--- PATIENT BILL DETAILS ---");
    printf("\nPatient ID        : %d", p.patientID);
    printf("\nTotal Bill        : ₹%.2f", totalBill);
    printf("\nSenior Discount   : ₹%.2f", discount);
    printf("\nInsurance Covered : ₹%.2f", insuranceCover);
    printf("\nFinal Amount Payable: ₹%.2f\n", finalBill);

    return 0;
}
```