1.Write a program to calculate sum of first n natural number.

```
#include<stdio.h>
int main()
{
   int n,sum=0;
   printf("enter natural number:");
   scanf("%d",&n);
   for (int i=0;i<=n;i++)
   {
      sum= sum+i;
   }
      printf("\n");
      printf("sum of natural number is=%d",sum);

   }
```

OUTPUT

```
enter natural number:10

sum of natural number is=55
```

2.Write a program to find the fibanocci series up to n terms

```
#include <stdio.h>

int main() {
   int n, first = 1, second = 1, next;

   printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci Series: ");
```

```
    for (int i = 1; i <= n; i++)
    {
        printf("%d ", first);
        next = first + second;
        first = second;
        second = next;
    }
    return 0;
}
```

OUTPUT

Enter the number of terms: 10

Fibonacci Series: 1 1 2 3 5 8 13 21 34 55

3)Write a program to reverse a number using for loop

```
 #include <stdio.h>

int main() {
    int num, rem = 0, rev = 0;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (int i = 1; num != 0; i++) {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    printf("Reversed number is: %d", rev);
    return 0;
```

```
}
```

OUTPUT

```
Enter a number: 1234

Reversed number is: 4321
```

4. Write a program to print pascal triangle row up to 8.

```
#include <stdio.h>


int main()
{
   int n, i, j, value;
   printf("Enter number of rows: ");
   scanf("%d", &n);
   for(i = 0; i < n; i++) {
      for(j = 0; j < n - i - 1; j++) {
         printf(" ");
      }
      value = 1;
      for(j = 0; j <= i; j++) {
         printf("%d ", value);
         value = value * (i - j) / (j + 1);
      }
      printf("\n");
   }
   return 0;
}
```

OUTPUT

```
Enter number of rows: 8
      1
```

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

5. Guess a number

```
#include <stdio.h>

int main() {
   int target = 2;
   int guess;
   int tries = 5;

   printf("Guess the number (not greater than 20):\n");

   for (int i = 1; i <= tries; i++) {
      printf("Attempt %d: ", i);
      scanf("%d", &guess);

      if (guess > 20) {
         printf("Warning: Please enter a number that is not greater than 20.\n");
         continue;
      }

      if (guess == target) {
         printf("Correct! The number is %d.\n", target);
```

```
            break;
        } else if (guess > target) {
            printf("Wrong! My number is less than %d.\n", guess);
        } else {
            printf("Wrong! My number is more than %d.\n", guess);
        }
    }

    if (guess != target) {
        printf("Sorry, you've used all attempts. The correct number was %d.\n", target);
    }

    return 0;
}
```

OUTPUT

Guess the number (not greater than 20):

Attempt 1: 12

Wrong! My number is less than 12.

Attempt 2: 8

Wrong! My number is less than 8.

Attempt 3: 4

Wrong! My number is less than 4.

Attempt 4: 2

Correct! The number is 2.

6. write a C program that prompts the user to enter a series of integers (up to a maximum of 20). The program should calculate and display the sum of all even numbers entered while skipping any negative numbers. Use the continue statement to skip processing for negative numbers.

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

```
int main() {
   int number, sum = 0;
   int count = 0;

   printf("Enter up to 20 integers (enter -1 to stop):\n");

   while (count < 20) {
      printf("Enter an integer: ");
      scanf("%d", &number);
      if (number == -1) {
         break;
      }
      if (number < 0) {
         continue;
      }
      if (number % 2 == 0) {
         sum += number;
      }

      count++;
   }
   printf("Sum of even numbers: %d\n", sum);
   return 0;
}
```

OUTPUT

```
Enter up to 20 integers (enter -1 to stop):
Enter an integer: 12
```

Enter an integer: 8

Enter an integer: -23

Enter an integer: 81

Enter an integer: 6

Enter an integer: 14

Enter an integer: -23

Enter an integer: -1

Sum of even numbers: 40

7. Problem Statement 1: Banking System Simulation

Description: Create a simple banking system simulation that allows users to create an
account, deposit money, withdraw money, and check their balance. The program should handle
multiple accounts and provide a menu-driven interface.

Requirements:
1. Use appropriate data types for account balance (e.g., float for monetary values) and user input (e.g., int for account numbers).

2. Implement a structure to hold account details (account number, account holder name, balance).

3. Use control statements to navigate through the menu options:

- **i. Create Account**
- **ii. Deposit Money**
- **iii. Withdraw Money**
- **iv. Check Balance**

4. Ensure that the withdrawal does not exceed the available balance and handle invalid inputs gracefully.

Example Input/Output:
Welcome to the Banking System
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 1

Enter account holder name: John Doe
Account created successfully! Account Number: 1001

Choose an option: 2
Enter account number: 1001
Enter amount to deposit: 500
Deposit successful! New Balance: 500.0

Choose an option: 3
Enter account number: 1001
Enter amount to withdraw: 200
Withdrawal successful! New Balance: 300.0

Choose an option: 4
Enter account number: 1001
Current Balance: 300.0

Choose an option: 5
Exiting the system.
has context menu
has context menu

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

#define MAX_ACCOUNTS 100
int accountNumbers[MAX_ACCOUNTS];
char accountHolders[MAX_ACCOUNTS][100];
float balances[MAX_ACCOUNTS];
int count = 0;

void createAccount();
void depositMoney();
void withdrawMoney();
void checkBalance();

int main() {
  int option;
  printf("Welcome to the Banking System\n");

  while (1) {
    printf("\n1. Create Account\n2. Deposit Money\n3. Withdraw Money\n4. Check
Balance\n5. Exit\nChoose an option: ");
    scanf("%d", &option);
```

```
        switch (option) {
            case 1: createAccount(); break;
            case 2: depositMoney(); break;
            case 3: withdrawMoney(); break;
            case 4: checkBalance(); break;
            case 5: printf("Exiting the system.\n"); return 0;
            default: printf("Invalid option. Please try again.\n");
        }
    }
}

void createAccount() {
    accountNumbers[count] = 1001 + count;
    printf("Enter account holder name: ");
    scanf("%s", accountHolders[count]);
    balances[count++] = 0.0;
    printf("Account created successfully! Account Number: %d\n",
accountNumbers[count - 1]);
}

void depositMoney() {
    int accNumber, found = 0;
    float amount;
    printf("Enter account number: ");
    scanf("%d", &accNumber);

    for (int i = 0; i < count && !found; i++) {
        if (accountNumbers[i] == accNumber) {
            printf("Enter amount to deposit: ");
            scanf("%f", &amount);
            if (amount > 0) {
                balances[i] += amount;
                printf("Deposit successful! New Balance: %.2f\n", balances[i]);
            } else {
                printf("Invalid amount. Deposit failed.\n");
            }
            found = 1;
        }
    }
    if (!found) printf("Account not found.\n");
}

void withdrawMoney() {
    int accNumber, found = 0;
    float amount;
    printf("Enter account number: ");
    scanf("%d", &accNumber);
```

```
   for (int i = 0; i < count && !found; i++) {
      if (accountNumbers[i] == accNumber) {
         printf("Enter amount to withdraw: ");
         scanf("%f", &amount);
         if (amount > 0 && amount <= balances[i]) {
            balances[i] -= amount;
            printf("Withdrawal successful! New Balance: %.2f\n", balances[i]);
         } else {
            printf(amount > balances[i] ? "Insufficient balance. Withdrawal failed.\n" :
"Invalid amount. Withdrawal failed.\n");
         }
         found = 1;
      }
   }
   if (!found) printf("Account not found.\n");
}

void checkBalance() {
   int accNumber, found = 0;
   printf("Enter account number: ");
   scanf("%d", &accNumber);

   for (int i = 0; i < count && !found; i++) {
      if (accountNumbers[i] == accNumber) {
         printf("Current Balance: %.2f\n", balances[i]);
         found = 1;
      }
   }
   if (!found) printf("Account not found.\n");
}
```

OUTPUT

Welcome to the Banking System

1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 1
Enter account holder name: MEGHA
Account created successfully! Account Number: 1001

1. Create Account
2. Deposit Money

3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 2
Enter account number: 1001
Enter amount to deposit: 500
Deposit successful! New Balance: 500.00

1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 3
Enter account number: 1001
Enter amount to withdraw: 200
Withdrawal successful! New Balance: 300.00

1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 4
Enter account number: 1001
Current Balance: 300.00

1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Choose an option: 5
Exiting the system.

8. Problem Statement 4: Weather Data Analysis

Description: Write a program that collects daily temperature data for a month and analyzes it to find the average temperature, the highest temperature, the lowest temperature, and how many days were above average.

Requirements:

1. Use appropriate data types (float for temperatures and int for days).

2. Store temperature data in an array.

3. Use control statements to calculate:

i. **Average Temperature of the month.**

ii. **Highest Temperature recorded.**

iii. **Lowest Temperature recorded.**

iv. **Count of days with temperatures above average.**

4. Handle cases where no data is entered.

Example Input/Output:

Enter temperatures for each day of the month (30 days):

Day 1 temperature: 72.5

Day 2 temperature: 68.0

...

Day 30 temperature: 75.0

Average Temperature of Month: XX.X

Highest Temperature Recorded: YY.Y

Lowest Temperature Recorded: ZZ.Z

Number of Days Above Average Temperature: N

```
#include <stdio.h>

int main() {
   int days = 30;
   float temperatures[30], sum = 0.0, average;
   float highest, lowest;
   int daysAboveAverage = 0;


   for (int i = 0; i < days; i++) {
      printf("Enter temperature for Day %d: ", i + 1);
      scanf("%f", &temperatures[i]);
      sum += temperatures[i];
```

```
      if (i == 0) {
         highest = temperatures[i];
         lowest = temperatures[i];
      } else {
         if (temperatures[i] > highest)
            highest = temperatures[i];
         if (temperatures[i] < lowest)
            lowest = temperatures[i];
      }
   }


   average = sum / days;


   for (int i = 0; i < days; i++) {
      if (temperatures[i] > average) {
         daysAboveAverage++;
      }
   }

   printf("\nAverage Temperature of Month: %.1f\n", average);
   printf("Highest Temperature Recorded: %.1f\n", highest);
   printf("Lowest Temperature Recorded: %.1f\n", lowest);
   printf("Number of Days Above Average Temperature: %d\n", daysAboveAverage);

   return 0;
}
```

OUTPUT

```
Enter temperature for Day 1: 25
Enter temperature for Day 2: 45
Enter temperature for Day 3: 15
Enter temperature for Day 4: 40
Enter temperature for Day 5: 62
Enter temperature for Day 6: 3
Enter temperature for Day 7: 41
Enter temperature for Day 8: 22
Enter temperature for Day 9: 24
Enter temperature for Day 10: 130
Enter temperature for Day 11: 120
Enter temperature for Day 12: 452
Enter temperature for Day 13: -23
Enter temperature for Day 14: -45
Enter temperature for Day 15: -62
Enter temperature for Day 16: -42
```

Enter temperature for Day 17: -75
Enter temperature for Day 18: 36
Enter temperature for Day 19: 100
Enter temperature for Day 20: 110
Enter temperature for Day 21: 160
Enter temperature for Day 22: 41
Enter temperature for Day 23: 162
Enter temperature for Day 24: 753
Enter temperature for Day 25: 32
Enter temperature for Day 26: 23
Enter temperature for Day 27: 14
Enter temperature for Day 28: 101
Enter temperature for Day 29: 105
Enter temperature for Day 30: 109

Average Temperature of Month: 82.6
Highest Temperature Recorded: 753.0
Lowest Temperature Recorded: -75.0
Number of Days Above Average Temperature: 11

9. Problem Statement : Inventory Management System

Description: Create an inventory management system that allows users to manage products in a store. Users should be able to add new products, update existing product quantities, delete products, and view inventory details.

Requirements:

1. Use appropriate data types for product details (e.g., char arrays for product names, int for quantities, float for prices).

2. Implement a structure to hold product information.

3. Use control statements for menu-driven operations:

i. **Add Product**

ii. **Update Product Quantity**

iii. **Delete Product**

iv. **View All Products in Inventory**

4. Ensure that the program handles invalid inputs and displays appropriate error messages.

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

#define MAX_PRODUCTS 100

int main() {
  char productNames[MAX_PRODUCTS][50];
  int productQuantities[MAX_PRODUCTS];
  float productPrices[MAX_PRODUCTS];
  int productCount = 0;
  int choice;

  while (1) {
    printf("\nInventory Management System\n");
    printf("1. Add Product\n");
    printf("2. Update Product Quantity\n");
    printf("3. Delete Product\n");
    printf("4. View All Products in Inventory\n");
    printf("5. Exit\n");
    printf("Choose an option: ");
    scanf("%d", &choice);

    if (choice == 1) {
      if (productCount >= MAX_PRODUCTS) {
        printf("Inventory is full. Cannot add more products\n");
      } else {
        printf("Enter product name: ");
        scanf("%s", productNames[productCount]);
        printf("Enter product quantity: ");
        scanf("%d", &productQuantities[productCount]);
        printf("Enter product price: ");
        scanf("%f", &productPrices[productCount]);
        productCount++;
        printf("Product added successfully\n");
      }
    } else if (choice == 2) {
      char name[50];
      int newQuantity, found = 0;

      printf("Enter product name to update quantity: ");
      scanf("%s", name);

      for (int i = 0; i < productCount; i++) {
        if (strcmp(productNames[i], name) == 0) {
          printf("Enter new quantity: ");
          scanf("%d", &newQuantity);
          productQuantities[i] = newQuantity;
```

```
            printf("Quantity updated successfully\n");
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Product not found\n");
    }
} else if (choice == 3) {
    char name[50];
    int found = 0;

    printf("Enter product name to delete: ");
    scanf("%s", name);

    for (int i = 0; i < productCount; i++) {
        if (strcmp(productNames[i], name) == 0) {
            for (int j = i; j < productCount - 1; j++) {
                strcpy(productNames[j], productNames[j + 1]);
                productQuantities[j] = productQuantities[j + 1];
                productPrices[j] = productPrices[j + 1];
            }
            productCount--;
            printf("Product deleted successfully\n");
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Product not found\n");
    }
} else if (choice == 4) {
    if (productCount == 0) {
        printf("Inventory is empty\n");
    } else {
        printf("Inventory Details:\n");
        for (int i = 0; i < productCount; i++) {
            printf("Product Name: %s Quantity: %d Price: %.2f\n",
                productNames[i], productQuantities[i], productPrices[i]);
        }
    }
} else if (choice == 5) {
    printf("Exiting the system\n");
    break;
} else {
    printf("Invalid choice. Please try again\n");
}
```

```
    }

    return 0;
}
```

OUTPUT

```
Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 1

Enter product name: Widget A
Enter product quantity: 50
Enter product price: 19.99
Product added successfully

Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 1

Enter product name: Widget B
Enter product quantity: 30
Enter product price: 29.99
Product added successfully

Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 4

Inventory Details:
Product Name: Widget A Quantity: 50 Price: 19.99
Product Name: Widget B Quantity: 30 Price: 29.99

Inventory Management System
1. Add Product
```

2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 2

Enter product name to update quantity: Widget A
Enter new quantity: 60
Quantity updated successfully

Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 4

Inventory Details:
Product Name: Widget A Quantity: 60 Price: 19.99
Product Name: Widget B Quantity: 30 Price: 29.99

Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 3

Enter product name to delete: Widget B
Product deleted successfully

Inventory Management System
1. Add Product
2. Update Product Quantity
3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 4

Inventory Details:
Product Name: Widget A Quantity: 60 Price: 19.99

Inventory Management System
1. Add Product
2. Update Product Quantity

3. Delete Product
4. View All Products in Inventory
5. Exit
Choose an option: 5

Exiting the system

10. Write a program to print the multiplication table from 1 to 10:

```
#include <stdio.h>

int main() {
   int num;

   printf("Multiplication Table from 1 to 10:\n");

   for (int num = 1; num <= 10; num++) {
      printf("Multiplication table of %d:\n", num);
      for (int i = 1; i <= 10; i++) {
         printf("%d * %d = %d\t", num, i, num * i);
      }
      printf("\n");
   }

   return 0;
}
```

OUTPUT

Multiplication Table from 1 to 10:
Multiplication table of 1:
1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 1 x 4 = 4 1 x 5 = 5 1 x 6 = 6 1 x 7 = 7 1 x 8 = 8 1 x 9 = 9 1 x 10 = 10
Multiplication table of 2:
2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20
Multiplication table of 3:
3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 3 x 4 = 12 3 x 5 = 15 3 x 6 = 18 3 x 7 = 21 3 x 8 = 24 3 x 9 = 27 3 x 10 = 30
Multiplication table of 4:
4 x 1 = 4 4 x 2 = 8 4 x 3 = 12 4 x 4 = 16 4 x 5 = 20 4 x 6 = 24 4 x 7 = 28 4 x 8 = 32 4 x 9 = 36 4 x 10 = 40
Multiplication table of 5:
5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50
Multiplication table of 6:

6 x 1 = 6 6 x 2 = 12 6 x 3 = 18 6 x 4 = 24 6 x 5 = 30 6 x 6 = 36 6 x 7 = 42 6 x 8 = 48 6 x 9 = 54 6 x 10 = 60
Multiplication table of 7:
7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70
Multiplication table of 8:
8 x 1 = 8 8 x 2 = 16 8 x 3 = 24 8 x 4 = 32 8 x 5 = 40 8 x 6 = 48 8 x 7 = 56 8 x 8 = 64 8 x 9 = 72 8 x 10 = 80
Multiplication table of 9:
9 x 1 = 9 9 x 2 = 18 9 x 3 = 27 9 x 4 = 36 9 x 5 = 45 9 x 6 = 54 9 x 7 = 63 9 x 8 = 72 9 x 9 = 81 9 x 10 = 90
Multiplication table of 10:
10 x 1 = 10 10 x 2 = 20 10 x 3 = 30 10 x 4 = 40 10 x 5 = 50 10 x 6 = 60 10 x 7 = 70 10 x 8 = 80 10 x 9 = 90 10 x 10 = 100

11. Write a program to print the pattern

```
*
* *
* * *
* * * *
* * * * *
```

```
#include <stdio.h>

int main() {
  int i = 1;
  int rows = 5;

  while (i <= rows) {
    int j = 1;

    while (j <= i) {
      printf("* ");
      j++;
    }

    printf("\n");
    i++;
  }

  return 0;
}
```

12.Write a program to print the pattern

```
  *
 * *
* * *
* * * *
* * * * *
```

```
#include <stdio.h>

int main() {
   int i = 1;
   int rows = 5;
   while (i <= rows) {

      int space = 1;
      while (space <= rows - i) {
         printf(" ");
         space++;
      }

      int j = 1;
      while (j <= i) {
         printf("* ");
         j++;
      }


      printf("\n");
      i++;
   }

   return 0;
}
```