Write a C program to accept an ID from the user and display the corresponding department.

-> IDs from 11 to 15 belong to the Software department.

-> IDs from 16 to 20 belong to the Developer department.

-> IDs from 21 to 23 belong to the Management department.

Sample-run :-

Test Cases :-

|---|

|Test Case No. Input ID Expected Output |

|---|

| 1 11 Software department |

| 2 16 Developer department |

| 3 21 Management department |

| 4 (Invalid) 25 Invalid ID |

|---|

```
#include <stdio.h>

int main() {
    int id;

    // Prompt the user for an ID
    printf("Enter your ID: ");
    scanf("%d", &id);

    // Switch statement to determine the department based on the ID
    switch(id) {
        case 11:
        case 12:
        case 13:
        case 14:
        case 15:
            printf("Department: Software\n");
            break;
        case 16:
        case 17:
        case 18:
        case 19:
        case 20:
```

```
            printf("Department: Developer\n");
            break;
        case 21:
        case 22:
        case 23:
            printf("Department: Management\n");
            break;
        default:
            printf("Invalid ID\n");
            break;
    }

    return 0;
}
```

|--------------|

| Program-2 :- | [ok]

|-

Write a C program to accept a user ID of 1001 and a password of 1010. Prompt the user to enter their ID. If the ID is valid, ask the user to enter their password. If the password is correct, display the name of the user. Otherwise, display "Incorrect Password". If the ID does not exist, display "Incorrect ID".

Develop this program using a switch-case statement.

Sample-run :-

Test-Case-1 (Valid)

Input :-

User ID: 1001

Password: 1010

Expected Output: Display the user's name (e.g., "User: John Doe").

Test-Case-2 (Invalid Password)

Input :-

User ID: 1001

Password: 2020

Expected Output: "Incorrect Password"

Test-Case-3 (Invalid ID)

Input :-

User ID: 2000

Expected Output: "Incorrect ID"

Test-Case-4 (Empty Input for ID)

Input :-

User ID: (no input)

Expected Output: "Incorrect ID"

```
// Online C compiler to run C program online
#include <stdio.h>

int main() {
 int id;
    printf("enter the id:");
    scanf("%d",&id);
    int password;

    if(id==1001)
```

```
    {

        printf("enter the password:");
        scanf("%d",&password);
      switch(password==1010)
      {
          case 1:
          printf("User: John Doe"); return 0;

         case 0:
         printf("Invalid password");
     }
    }
    else
    {
       printf("Incorrect ID");
    }
    return 0;
}
```

or

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

```
int main() {
    int user_id, password;

    // Predefined valid ID and password
    const int valid_id = 1001;
    const int correct_password = 1010;

    // Prompt the user for the ID
    printf("User ID: ");
    scanf("%d", &user_id);

    // Switch-case to handle different user ID input scenarios
    switch (user_id) {
        case 1001: // Valid ID case
            printf("Password: ");
            scanf("%d", &password);

            switch (password) {
                case 1010:  // Correct password case
                    printf("User: John Doe\n");
                    break;
                default:  // Incorrect password case
```

```
            printf("Incorrect Password\n");
            break;
        }
        break;

    default:  // Invalid ID case
        printf("Incorrect ID\n");
        break;
  }

  return 0;
}
```

|--------------|

| **Program-3 :-** | **[ok]**

|--------------|

Write a C program that asks for a person's name and their game score. Then, it asks for the second person's name and score. The program will print the winner's name and display by how many points the winner won, by comparing the scores. Develop this program using a switch case.

Sample-run :-

Test Case 1: Valid Input (Player 1 Wins)

Input :-

Enter name of Player 1: Alice

Enter score of Player 1: 85

Enter name of Player 2: Bob

Enter score of Player 2: 75

Expected Output :-

Winner: Alice

Points difference: 10

Test Case 2: Valid Input (Player 2 Wins)

Input :-

Enter name of Player 1: David

Enter score of Player 1: 60

Enter name of Player 2: Eve

Enter score of Player 2: 95

Expected Output :-

Winner: Eve

Points difference: 35

Test Case 3: Valid Input (Tie Case)

Input :-

Enter name of Player 1: John

Enter score of Player 1: 80

Enter name of Player 2: Sarah

Enter score of Player 2: 80

Expected Output :-

It's a tie!

Points difference: 0

Test Case 4: Invalid Input (Negative Score)

Input :-

Enter name of Player 1: Chris

Enter score of Player 1: -50

Enter name of Player 2: Anna

Enter score of Player 2: 70

Expected Output :-

Invalid input: Score cannot be negative.

```
#include <stdio.h>

int main() {
    char p1[50], p2[50];
    int ps1, ps2;



    printf("Enter name of Player 1: ");
    scanf("%s", p1);
    printf("Enter score of Player 1: ");
    scanf("%d", &ps1);
```

```
// Input for Player 2
printf("Enter name of Player 2: ");
scanf("%s", p2);
printf("Enter score of Player 2: ");
scanf("%d", &ps2);


if (ps1 < 0 || ps2 < 0) {
    printf("Invalid input: Score cannot be negative.\n");
    return 0;
}

    switch (1) {
    case 1:
        if (ps1 > ps2) {
            printf("Winner: %s\n", p1);
            printf("Points difference: %d\n", ps1 - ps2);
        } else if (ps2 > ps1) {
            printf("Winner: %s\n", p2);
            printf("Points difference: %d\n", ps2 - ps1);
        } else {
            printf("It's a tie!\n");
```

```
            printf("Points difference: 0\n");
        }
        break;
    }

    return 0;
}
```

|----------|

| Program-4 :- | [ok]

|----------|

Using a switch statement, write a C program to take input marks of five subjects from the user: Physics, Chemistry, Biology, Mathematics, and Computer.

Calculate the percentage and determine the grade according to the following criteria:

Percentage >= 90%: Grade A

Percentage >= 80%: Grade B

Percentage >= 70%: Grade C

Percentage >= 60%: Grade D

Percentage >= 40%: Grade E

Percentage < 40%: Grade F

Note: Take all input values as integers.

Sample-run :-

Test Case 1 (Valid Input) :-

Input marks: Physics = 85, Chemistry = 90, Biology = 88, Mathematics = 92, Computer = 80

Total Marks: 435

Percentage: 87%

-> Expected Grade: B

Test Case 2 (Valid Input) :-

Input marks: Physics = 45, Chemistry = 50, Biology = 55, Mathematics = 60, Computer = 40

Total Marks: 250

Percentage: 50%

-> Expected Grade: E

Test Case 3 (Valid Input) :-

Input marks: Physics = 95, Chemistry = 92, Biology = 90, Mathematics = 93, Computer = 96

Total Marks: 466

Percentage: 93.2%

-> Expected Grade: A

Test Case 4 (Invalid Input) :-

Input marks: Physics = -10, Chemistry = 110, Biology = 80, Mathematics = 95, Computer = 70

Reason: Marks cannot be negative or exceed 100.

-> Expected Output: Display an error message indicating invalid input.

```
#include<stdio.h>
int main()
```

```
{
   int s1,s2,s3,s4,s5,totm,per,Grade;

   printf("enter  marks for physics:");
   scanf("%d",&s1);
   printf("enter  marks for chemistry:");
   scanf("%d",&s2);
   printf("enter  marks for Bilology:");
   scanf("%d",&s3);
   printf("enter  marks for Mathematics:");
   scanf("%d",&s4);
   printf("enter  marks for computer science:");
   scanf("%d",&s5);

   if (s1 < 0 || s1 > 100 || s2 < 0 || s2 > 100 || s3 < 0 || s3 > 100 || s4 < 0 || s4 > 100 ||
s5 < 0 || s5 > 100) {
      printf("Error: Marks cannot be negative or exceed 100.\n");
      return 1;

   totm=s1+s2+s3+s4+s5;
   printf("total marks=%d/500\n",totm);
   per=totm*100/500;
```

```
printf("percentage =%.d\n",per);


 switch(per/10) {
    case 10:
    case 9:
       printf("Grade A\n");
       break;
    case 8:
       printf("Grade B\n");
       break;
    case 7:
       printf("Grade C\n");
       break;
    case 6:
       printf("Grade D\n");
       break;
    case 5:
       printf("Grade E\n");
       break;
    default:
       printf("Grade F\n");
       break;
 }
```

}

return 0;

}

|--------------|

| Program-5 :- |

|--------------|

Write a C program that reads a float value and a character from the console. The character can be 'd' for deposit or 'w' for withdrawal.

Start with a minimum balance of 2000.

-> Prompt the user to initialize the minimum balance. Do not allow 0 or negative values. If the balance is 0 or negative, print: "Amount can't be stored".

-> If the user chooses the deposit operation, ask how much amount they want to deposit. Add that amount to the balance and print the updated balance.

-> If the user chooses the withdrawal operation, ask how much amount they want to withdraw. Subtract that amount from the balance and print the updated balance.

[Note]

-> If the deposit amount is negative or 0, print: "Invalid amount".

-> If the withdrawal amount is negative, 0, or greater than the current balance, print: "Invalid amount".

Hints :-

-> Declare variables for amount (float), balance (float), and transaction code (char).

-> Check the transaction code. If it is ‘d’ (deposit), ensure the amount is not negative.

If the amount is valid, update the balance:

balance = balance + amount;

-> Check the transaction code for withdrawal, ‘w’. If the code is ‘w’, subtract the withdrawn amount to update the balance:

balance = balance - amount;

Print the updated balance.

Sample-run :-

Test-Case-1: [Valid Deposit Operation]

Input :-

Enter initial balance: 3000

Enter transaction code (d for deposit, w for withdrawal): d

Enter amount: 500

Expected Output :-

Balance after deposit: 3500

Description: The user starts with an initial balance of 3000 and deposits 500. The new balance is updated and displayed correctly.

Test-Case-2: [Invalid Initial Balance]

Input :-

Enter initial balance: -100

Expected Output :-

Amount can't be stored

Description: The user tries to initialize the balance with a negative value. The program handles this by displaying an appropriate error message.

Test-Case-3: [Invalid Deposit Amount]

Input :-

Enter initial balance: 2000

Enter transaction code (d for deposit, w for withdrawal): d

Enter amount: -200

Expected Output :-

Description: The user selects the deposit operation but tries to deposit a negative amount. The program detects this invalid input and displays an error message.

Test-Case-4: [Invalid Withdrawal Amount]

Input :-

Enter initial balance: 2500

Enter transaction code (d for deposit, w for withdrawal): w

Enter amount: 3000

Expected Output :-

Invalid amount

```
#include <stdio.h>

int main() {
    float balance, amount;
    char transaction_code;

    // Prompt for initial balance
    printf("Enter initial balance: ");
    scanf("%f", &balance);

    // Check if the initial balance is valid (greater than 0)
    if (balance <= 0) {
        printf("Amount can't be stored\n");
        return 0; // Exit the program if balance is invalid
```

```
    }

    // Prompt for transaction type (deposit or withdrawal)
    printf("Enter transaction code (d for deposit, w for withdrawal): ");
    scanf(" %c", &transaction_code);  // Added space before %c to consume any leftover newline character

    // Switch-case to handle different transaction codes
    switch (transaction_code) {
        case 'd': // Deposit operation
            printf("Enter amount: ");
            scanf("%f", &amount);

            // Check if the deposit amount is valid (greater than 0)
            if (amount <= 0) {
                printf("Invalid amount\n");
            } else {
                balance += amount;
                printf("Balance after deposit: %.2f\n", balance);
            }
            break;

        case 'w': // Withdrawal operation
```

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

            // Check if the withdrawal amount is valid (greater than 0 and less than or equal to current balance)
            if (amount <= 0 || amount > balance) {
                printf("Invalid amount\n");
            } else {
                balance -= amount;
                printf("Balance after withdrawal: %.2f\n", balance);
            }
            break;

        default:
            printf("Invalid transaction code\n");
            break;
    }

    return 0;
}
```