Exercício 1:

Defina as classes de equivalência e os casos de testes para um componente que aceita uma cor como entrada (retornando o nome da cor de acordo com o código passado)

1 - RED

2 – YELLOW

3 – BLUE

4 – VIOLET

Passo 1: definições das restrições de entrada

Código da Cor: **Integer**

Passo 2: definição das classes válidas

1. Código da cor não pode ser vazio;
2. Código deve estar entre 1 e 4;
3. Código não deve ter mais de um dígito;

Passo 3: definição das classes inválidas

1. Código em branco;
2. Código com mais de 1 dígito;
3. Código inferior a 1;
4. Código maior do que 4;

Passo 4: criar tabela de classes de equivalência

Restrições de Entrada	Classes válidas	Classes inválidas
Código (c)	● c não vazio; ● 1 >= c <= 4; ● c não deve possuir mais de 1 dígito;	● c vazio; ● c com mais de 1 dígito; ● c < 1; ● c > 4;

Passo 5: definir casos de teste

1. Código 1: **Válido**
2. Código 0: **Inválido**
3. Código 5: **Inválido**
4. Código “”: **Inválido**

5. Código 11: **Inválido**

Exercício 2:

Defina as classes de equivalência e os valores limites e crie os casos de testes para o seguinte módulo: Este módulo é parte de um sistema de TV por assinatura. O módulo recebe o pagamento dos assinantes a partir de R$ 0,01 à R$ 99.999,00. Além disto, este módulo recebe o status do assinante (que pode ser: regular, estudante/aposentado ou VIP). Ao final ele indica um código: 0 – Sucesso 1 – erro no valor 2 – status incorreto

Passo 1: definições das restrições de entrada

1. Pagamento: **Double**
2. Status: **String**

Passo 2: definições das classes válidas

1. Pagamento não vazio;
2. Pagamento >= R$ 0,01 e Pagamento <= R$ 99.999,00;
3. Status não vazio;
4. Status = “regular”;
5. Status = “estudante”;
6. Status = “aposentado”;
7. Status = “VIP”;

Passo 3: definições das classes inválidas

1. Pagamento vazio;
2. Pagamento < R$ 0,01;
3. Pagamento > R$ 99.999,00;
4. Status vazio;
5. Status != (“regular”, “estudante”, “aposentado”, “VIP”);

Passo 4: criar tabela de classes de equivalência

Restrições de Entrada	Classes válidas	Classes inválidas
Pagamento (p);	● p não vazio; ● R$ 0,01 >= p <= R$ 99.999,00	● p vazio; ● p < R$ 0,01; ● p > R$ 99.999,00;
Status (s);	● s não vazio; ● s = “regular”; ● s = “estudante”; ● s = “aposentado”; ● s = “VIP”;	● s vazio; ● s != (“regular”, “estudante”, “aposentado”, “VIP”)

Passo 5: definição dos casos de teste

1. Pagamento = R$ 0,01 e Status = “regular”: **0**
2. Pagamento = R$ 10,01 e Status = “estudante”: **0**
3. Pagamento = R$ 110,01 e Status = “aposentado”: **0**
4. Pagamento = R$ 999,01 e Status = “VIP”: **0**
5. Pagamento = R$ 0,00 e Status = “regular”: **1**
6. Pagamento = R$ 0,01 e Status = “”: **2**
7. Pagamento = R$ 99.999,01 e Status = “regular”: **1**
8. Pagamento = R$ 0,01 e Status = “chefe”: **2**

Exercício 3:

A classe Aluno possui um método: String situacao(double media, boolean temG2) Este método é usado para definir a situação do aluno (“aprovado”, “G2” ou “rep”). Os parâmetros são:

(a) média G1; (lembre-se que a nota máxima de um aluno é 10)

(b) tipo da disciplina (booleano: true se a disciplina tem G2, false em caso contrário) Se a disciplina tem G2, a situação é “aprovado” se G1 >= 7, “rep” se G1 =< 4 e “G2” caso contrário. Se a disciplina não tem G2, a situação é “aprovado” se G1 >= 5 e “rep” caso contrário.

Apresente os casos de testes aplicando as técnicas de Classes de Equivalência, Valor Limite e grafo causa-efeito

Passo 1: definição das restrições de entrada

1. Média: **Double**
2. TemG2: **Boolean**

Passo 2: definições das classes válidas

1. Média não vazia;
2. Média >= 0,0 && Média <= 10,0;
3. TemG2 não vazio;

Passo 3: definições das classes inválidas

1. Média vazia;
2. Média < 0,0;
3. Média > 10,0;

Passo 4: criação da tabela das classes de equivalência

Restrições de Entrada	Classes válidas	Classes inválidas
Média (m)	● m não vazio; ● m >= 0,0 && m <= 10,0	● m vazio; ● m < 0,0; ● m >10,0;
TemG2 (t);	● t não vazio;	● t vazio;

Passo 5: criação dos casos de teste

1. Média = 4,0 e TemG2 = true : retorna G2;
2. Média = 3,5 e TemG2 = true: retorna reprovado;
3. Média = 4,0 e TemG2 = false : retorna reprovado;
4. Média = 5,0 e TemG2 = false: retorna aprovado;
5. Média = 7,0 e TemG2 = true : retorna aprovado;
6. Média = 7,0 e TemG2 = false: retorna aprovado;
7. Média = -1,5 e TemG2 = true: **Inválido**;
8. Média = 11,3 e TemG2 = false: **Inválido**;
9. Média = "" e TemG2 = false: **Inválido;**
10. Média = 10,0 e TemG2 = "": **Inválido;**

Exercício 4:

Imprime mensagens Um programa lê dois caracteres e, de acordo com eles, mensagens serão impressas da seguinte forma: o primeiro caracter deve ser um ‘A’ ou um ‘B’. O segundo caracter deve ser um dígito. Nessa situação, o arquivo deve ser atualizado. Se o primeiro caracter estiver incorreto, enviar a mensagem ‘X’. Se o segundo caracter estiver incorreto, enviar a mensagem ‘Y’.

Passo 1: definições das restrições de entrada

1. Caractere 1: **Char;**
2. Caractere 2 : **Int;**

Passo 2: definição das classes válidas

1. Caractere 1 Deve ser A ou B;
2. Caractere 1 não vazio;
3. Caractere 2 não vazio;
4. Caractere 2 deve ser um dígito (0 - 9);

5. Caractere 2 >=0 && Caractere 2 <= 9;

Passo 3: definição das classes inválidas

1. Caractere 1 vazio;
2. Caractere 1 diferente de A ou B;
3. Caractere 2 vazio;
4. Caractere 2 > 9;
5. Caractere 2 < 0;

Passo 4: criar tabela de classes de equivalência

Restrições de Entrada	Classes válidas	Classes inválidas
Caractere 1 (c1)	● c1 não vazio; ● c1 = A; ● c1 = B;	● c1 vazio; ● c1 != A; ● c1 != B;
Caractere 2 (c2)	● c2 não vazio; ● c2 >= 0 && c2 <= 9;	● c2 vazio; ● c2 < 0; ● c2 > 9;

Passo 5: definir casos de teste

1. C1 = A e C2 = 0 : **Válido;**
2. C1 = B e C2 = 9 : **Válido;**
3. C1 = C e C2 = 9 : X**;**
4. C1 = B e C2 = 10 : Y**;**
5. C1 = '' e C2 = 9 : X**;**
6. C1 = A e C2 = '' : Y**;**