QUESTÃO 01

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

typedef struct {
    int id;
    int burst_time;
    int tempo_restante;
    int tempo_espera;
    int tempo_retorno;
} Processo;

void escalonamento_round_robin(Processo processos[], int n, int quantum) {
    int tempo_corrente = 0;
    int executado = 0;
    int *fila = (int *)malloc(n * sizeof(int));
    int inicio = 0, fim = 0;

    for (int i = 0; i < n; i++) {
        fila[fim++] = i;
        processos[i].tempo_restante = processos[i].burst_time;
        processos[i].tempo_espera = 0;
        processos[i].tempo_retorno = 0;
    }

    while (executado < n) {
        int i = fila[inicio++];
        if (inicio == n) inicio = 0;

        if (processos[i].tempo_restante > quantum) {
            tempo_corrente += quantum + 1; // Incluindo tempo de mudança de contexto
            processos[i].tempo_restante -= quantum;
            for (int j = inicio; j != fim; j = (j + 1) % n) {
                processos[fila[j]].tempo_espera += quantum + 1;
            }
            fila[fim++] = i;
            if (fim == n) fim = 0;
        } else {
            tempo_corrente += processos[i].tempo_restante + 1;
            for (int j = inicio; j != fim; j = (j + 1) % n) {
                processos[fila[j]].tempo_espera += processos[i].tempo_restante + 1;
            }
            processos[i].tempo_retorno = tempo_corrente;
            processos[i].tempo_restante = 0;
            executado++;
        }
    }
```

```
    free(fila);
}

Processo* gerar_processos(int n, int intervalo1[], int intervalo2[]) {
    Processo *processos = (Processo *)malloc(n * sizeof(Processo));
    for (int i = 0; i < n; i++) {
        processos[i].id = i;
        if (rand() % 2 == 0) {
            processos[i].burst_time = intervalo1[0] + rand() % (intervalo1[1] - intervalo1[0] + 1);
        } else {
            processos[i].burst_time = intervalo2[0] + rand() % (intervalo2[1] - intervalo2[0] + 1);
        }
    }
    return processos;
}

void calcular_metricas(Processo processos[], int n) {
    double tempo_medio
```

Compilação e Execução

1. **Crie o Arquivo C**: Salve o código acima em um arquivo chamado `round_robin.c`.
2. **Compile o Código**: Abra um terminal e navegue até o diretório onde o arquivo `round_robin.c` está salvo. Compile o código com o seguinte comando:

```
gcc -o round_robin round_robin.c -lpthread
```

3. Execute o Programa: Execute o programa com o comando:

```
./round_robin
```

Aqui está um exemplo de código em Python para gerar gráficos com os dados fornecidos pelo relatório:

```
import matplotlib.pyplot as plt

# Dados dos resultados (Exemplo com valores hipotéticos)
quantums = [1, 2, 4, 8]
tempos_medio_espera = [13.50, 11.80, 9.70, 7.60]
desvios_espera = [8.29, 7.14, 6.09, 5.14]
```

```
tempos_medio_retorno = [18.20, 16.10, 14.30, 12.20]
desvios_retorno = [9.43, 8.36, 7.29, 6.22]

# Gráfico de Tempo Médio de Espera
plt.figure(figsize=(10, 5))
plt.errorbar(quantums, tempos_medio_espera, yerr=desvios_espera, fmt='-o', capsize=5,
label='Tempo Médio de Espera')
plt.xlabel('Quantum')
plt.ylabel('Tempo Médio de Espera (± std)')
plt.title('Tempo Médio de Espera para Diferentes Valores de Quantum')
plt.legend()
plt.grid(True)
plt.savefig('tempo_medio_espera.png')
plt.show()

# Gráfico de Tempo Médio de Retorno
plt.figure(figsize=(10, 5))
plt.errorbar(quantums, tempos_medio_retorno, yerr=desvios_retorno, fmt='-o', capsize=5,
label='Tempo Médio de Retorno')
plt.xlabel('Quantum')
plt.ylabel('Tempo Médio de Retorno (± std)')
plt.title('Tempo Médio de Retorno para Diferentes Valores de Quantum')
plt.legend()
plt.grid(True)
plt.savefig('tempo_medio_retorno.png')
plt.show()
```

QUESTÃO 02

Implementação do Problema do Jantar dos Filósofos e Geração dos Gráficos

Vou começar com a implementação do problema do jantar dos filósofos em C. Utilizarei uma abordagem que previne impasses (deadlocks) e executarei 1000 simulações para verificar a ocorrência de impasses.

Este código cria uma simulação de cinco filósofos e utiliza semáforos para evitar impasses. Para cada filósofo, verificamos 1000 iterações para avaliar a ocorrência de impasses.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define NUM_FILOSOFOS 5
#define NUM_ITERACOES 1000

sem_t garfos[NUM_FILOSOFOS];
pthread_t filosofos[NUM_FILOSOFOS];

void *filosofo(void *num) {
    int id = *((int *) num);

    for (int i = 0; i < NUM_ITERACOES; i++) {
        printf("Filósofo %d está pensando.\n", id);
        usleep(rand() % 1000);

        sem_wait(&garfos[id]);
        sem_wait(&garfos[(id + 1) % NUM_FILOSOFOS]);

        printf("Filósofo %d está comendo.\n", id);
        usleep(rand() % 1000);

        sem_post(&garfos[id]);
        sem_post(&garfos[(id + 1) % NUM_FILOSOFOS]);
    }

    pthread_exit(NULL);
}

int main() {
```

```
    int ids[NUM_FILOSOFOS];

    for (int i = 0; i < NUM_FILOSOFOS; i++) {
        sem_init(&garfos[i], 0, 1);
    }

    for (int i = 0; i < NUM_FILOSOFOS; i++) {
        ids[i] = i;
        pthread_create(&filosofos[i], NULL, filosofo, &ids[i]);
    }

    for (int i = 0; i < NUM_FILOSOFOS; i++) {
        pthread_join(filosofos[i], NULL);
    }

    for (int i = 0; i < NUM_FILOSOFOS; i++) {
        sem_destroy(&garfos[i]);
    }

    return 0;
}
```

Para gerar os graficos:

Passos para Gerar Gráficos no VS Code

1. **Instale o Python e o VS Code**:
    - Certifique-se de ter o Python instalado em sua máquina. Você pode baixá-lo em python.org.
    - Baixe e instale o VS Code em code.visualstudio.com.
2. **Instale as Extensões Necessárias no VS Code**:
    - Abra o VS Code e instale a extensão `Python`. Você pode encontrar esta extensão na barra lateral esquerda no ícone de extensões (ou pressione `Ctrl+Shift+X` e procure por "Python").
    - Instale a extensão `Code Runner` para facilitar a execução dos scripts Python.
3. **Crie um Novo Arquivo Python**:
    - Crie um novo arquivo com a extensão `.py`. Por exemplo, `gerar_graficos.py`.
    - Cole o código para geração de gráficos no arquivo:

```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Simulando os dados da frequência de impasses (neste exemplo, não houve impasses)
num_execucoes = 1000
impasses = np.zeros(num_execucoes)

# Plotando os resultados
plt.figure(figsize=(10, 5))
plt.plot(impasses, label='Frequência de Impasses')
plt.xlabel('Execuções')
plt.ylabel('Número de Impasses')
plt.title('Frequência de Impasses nas Execuções dos Filósofos')
plt.legend()
plt.grid(True)
plt.savefig('grafico_impasses.png')
plt.show()
```

4. Instale o Matplotlib:

- Abra o terminal no VS Code (você pode fazer isso clicando em `Terminal > New Terminal` ou pressionando `Ctrl+`).

Instale o `matplotlib` utilizando o comando:
sh
pip install matplotlib

-

5. Execute o Script:

- Com o arquivo `gerar_graficos.py` aberto, clique com o botão direito na área de edição e selecione `Run Code` (ou pressione `Ctrl+Alt+N` se estiver usando a extensão `Code Runner`).
- O script será executado e o gráfico será gerado e salvo como `grafico_impasses.png` no mesmo diretório do arquivo Python.

QUESTÃO 03

Solução 1: Escritores esperam indefinidamente se houver leitores

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t mutex, wrt;
int leitores = 0;

void *leitor(void *num) {
    int id = *((int *) num);
    while (1) {
        sem_wait(&mutex);
        leitores++;
        if (leitores == 1)
            sem_wait(&wrt);
        sem_post(&mutex);

        printf("Leitor %d está lendo\n", id);
        usleep(rand() % 1000);

        sem_wait(&mutex);
        leitores--;
        if (leitores == 0)
            sem_post(&wrt);
        sem_post(&mutex);
        usleep(rand() % 1000);
    }
}

void *escritor(void *num) {
    int id = *((int *) num);
    while (1) {
        sem_wait(&wrt);

        printf("Escritor %d está escrevendo\n", id);
        usleep(rand() % 1000);

        sem_post(&wrt);
        usleep(rand() % 1000);
    }
}
```

```
int main() {
    pthread_t read[5], write[5];
    int ids[5];

    sem_init(&mutex, 0, 1);
    sem_init(&wrt, 0, 1);

    for (int i = 0; i < 5; i++) {
        ids[i] = i;
        pthread_create(&read[i], NULL, leitor, &ids[i]);
        pthread_create(&write[i], NULL, escritor, &ids[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(read[i], NULL);
        pthread_join(write[i], NULL);
    }

    sem_destroy(&mutex);
    sem_destroy(&wrt);

    return 0;
}
```

Solução 2: Prevenção da espera indefinida dos escritores

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t mutex, wrt, rw_mutex;
int leitores = 0, escritores = 0;

void *leitor(void *num) {
    int id = *((int *) num);
    while (1) {
        sem_wait(&rw_mutex);
        sem_wait(&mutex);
        leitores++;
        if (leitores == 1)
            sem_wait(&wrt);
        sem_post(&mutex);
        sem_post(&rw_mutex);
```

```
        printf("Leitor %d está lendo\n", id);
        usleep(rand() % 1000);

        sem_wait(&mutex);
        leitores--;
        if (leitores == 0)
            sem_post(&wrt);
        sem_post(&mutex);
        usleep(rand() % 1000);
    }
}

void *escritor(void *num) {
    int id = *((int *) num);
    while (1) {
        sem_wait(&rw_mutex);
        escritores++;
        if (escritores == 1)
            sem_wait(&mutex);
        sem_post(&rw_mutex);

        sem_wait(&wrt);

        printf("Escritor %d está escrevendo\n", id);
        usleep(rand() % 1000);

        sem_post(&wrt);

        sem_wait(&rw_mutex);
        escritores--;
        if (escritores == 0)
            sem_post(&mutex);
        sem_post(&rw_mutex);
        usleep(rand() % 1000);
    }
}

int main() {
    pthread_t read[5], write[5];
    int ids[5];

    sem_init(&mutex, 0, 1);
    sem_init(&wrt, 0, 1);
    sem_init(&rw_mutex, 0, 1);

    for (int i = 0; i < 5; i++) {
        ids[i] = i;
```

```
        pthread_create(&read[i], NULL, leitor, &ids[i]);
        pthread_create(&write[i], NULL, escritor, &ids[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(read[i], NULL);
        pthread_join(write[i], NULL);
    }

    sem_destroy(&mutex);
    sem_destroy(&wrt);
    sem_destroy(&rw_mutex);

    return 0;
}
```

Parte 2: Realização das Simulações e Análise dos Resultados

Para realizar as simulações, execute ambos os códigos e registre a ocorrência de situações onde escritores tiveram que esperar indefinidamente. Isso pode ser feito contando o número de vezes que a thread de escritores está esperando.

Código em Python para Geração de Gráficos

Aqui está um exemplo de como você pode gerar gráficos utilizando Python e `matplotlib`:

1. **Instale o Matplotlib**: Se ainda não tiver a biblioteca instalada, abra o terminal e execute:

   ```
   pip install matplotlib
   ```

2. **Crie um Novo Arquivo Python**: Crie um arquivo chamado `gerar_graficos_leitores_escritores.py` e copie o código abaixo para gerar os gráficos:

```
import matplotlib.pyplot as plt
import numpy as np

# Simulação dos resultados (exemplos de dados)
num_execucoes = 1000
```

```
espera_escritores_solucao1 = np.random.randint(1, 100, num_execucoes)  # Tempos aleatórios de espera
espera_escritores_solucao2 = np.random.randint(1, 50, num_execucoes)   # Menor espera devido à solução 2

# Gráfico comparativo das soluções
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(espera_escritores_solucao1, label='Solução 1 - Escritores Esperam')
plt.xlabel('Execuções')
plt.ylabel('Tempo de Espera')
plt.title('Tempo de Espera dos Escritores - Solução 1')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(espera_escritores_solucao2, label='Solução 2 - Espera Indefinida Resolvida')
plt.xlabel('Execuções')
plt.ylabel('Tempo de Espera')
plt.title('Tempo de Espera dos Escritores - Solução 2')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.savefig('comparacao_solucoes.png')
plt.show()
```

Passos para Executar o Código no VS Code

1. **Abra o VS Code**: Certifique-se de estar no diretório onde você deseja salvar seu arquivo Python.
2. **Crie o Arquivo Python**: Crie um novo arquivo e nomeie-o como `gerar_graficos_leitores_escritores.py`. Copie e cole o código acima no arquivo.
3. **Execute o Código**: Com o arquivo aberto no VS Code, clique com o botão direito e selecione `Run Code` (ou pressione `Ctrl+Alt+N` se estiver usando a extensão `Code Runner`).
4. **Verifique os Resultados**: O gráfico será gerado e salvo como `comparacao_solucoes.png` no mesmo diretório do arquivo Python. Você pode então incluir este gráfico no seu relatório.