INSTITUTO NACIONAL DE PESQUISA ESPACIAL – INPE

PROVA DISCURSIVA

TG19

DESENVOLVIMENTO DE SOFTWARE EMBARCADO

SUA PROVA

- Além deste caderno contendo **5 (cinco)** questões discursivas **com as respectivas folhas de rascunho**, você receberá do fiscal de prova as folhas de textos definitivos;

TEMPO

- Você dispõe de **4 (quatro) horas** para a realização da prova;
- **2 (duas) horas** após o início da prova, é possível retirar-se da sala, sem levar o caderno de questões;
- A partir dos **30 (trinta) minutos** anteriores ao término da prova é possível retirar-se da sala **levando o caderno de questões**.

NÃO SERÁ PERMITIDO

- Qualquer tipo de comunicação entre os candidatos durante a aplicação da prova;
- Anotar informações relativas às respostas em qualquer outro meio que não seja no caderno de questões e nas folhas de textos definitivos;
- Levantar da cadeira sem autorização do fiscal de sala;
- Usar o sanitário ao término da prova, após deixar a sala.

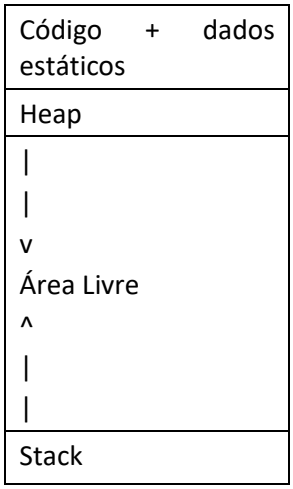
INFORMAÇÕES GERAIS

- Verifique se seu caderno de questões está completo, sem repetição de questões ou falhas. Caso contrário, **notifique imediatamente o fiscal da sala**, para que sejam tomadas as devidas providências;
- Confira seus dados pessoais, especialmente nome, número de inscrição e documento de identidade e leia atentamente as instruções para preencher as folhas de textos definitivos;
- Para o preenchimento das folhas de textos definitivos, use somente caneta esferográfica, fabricada em material transparente, com tinta preta ou azul;
- Assine seu nome apenas no(s) espaço(s) reservado(s) no cartão de respostas;
- Caso você tenha recebido caderno de cargo **diferente** do impresso em suas folhas de textos definitivos, o fiscal deve ser **obrigatoriamente** informado para o devido registro na ata da sala;
- O preenchimento das folhas de textos definitivos é de sua responsabilidade e **não será permitida a troca de folha de texto definitivo em caso de erro cometido pelo candidato**;
- Para fins de avaliação, serão levadas em consideração apenas os textos das folhas de textos definitivos;
- A FGV coletará as impressões digitais dos candidatos na lista de presença;
- Os candidatos serão submetidos ao sistema de detecção de metais quando do ingresso e da saída de sanitários durante a realização das provas.
- **Boa prova!**

FGV CONHECIMENTO

Questão 1

Sistemas Operacionais (SO) criam processos para executar programas. Inicialmente o SO carrega o código do programa e dados estáticos no espaço de endereços do processo. A figura a seguir é uma abstração do espaço de endereços do processo, contendo 4 blocos: código, Heap, área livre e Stack. Conforme o processo executa o código, a Heap e a Stack crescem e/ou diminuem de tamanho usando a área livre.



Defina Heap.

Defina um "memory leak" no contexto da execução de um processo e explique por que o "memory leak" pode causar degradações no desempenho de um SO.

Defina Stack.

Escreva duas linhas de código que, ao substituir as linhas de comentários (//A e //B), farão com que a execução do código a seguir sempre acarrete o esgotamento da área livre do processo, seja por parte da Heap ou por parte da Stack.

```
int SubA(int x)
{
	x = x * 1000;
	//A
}
int main(int argc, char* argv[])
{
	int op = 0;
	printf("\nopcao:");
	scanf("%d", &op);

	if (op == 1)
	{
		int s = SubA(10);	// CRASHHHHH……
	}
	else if (op == 2)
	{
		int j = 10;
		for (int i = 0; i < j;  j++)
		{
			char* buf;
			//B;
			if (buf == 0)
			{
				printf("BUMMMMMM");
				break;
			}
		}
	}
	printf("saindo");
	return 0;
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Questão 2

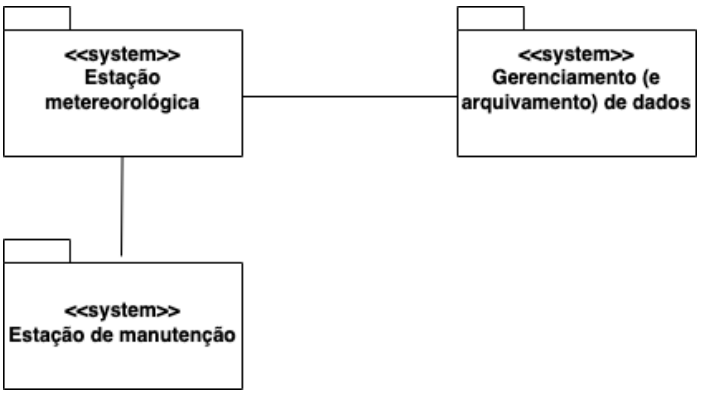
O desenvolvimento de software é um componente crítico da exploração espacial moderna, desempenhando papel vital na concepção, desenvolvimento e operação de naves espaciais e satélites, permitindo que cientistas e engenheiros monitorizem e controlem estes sistemas complexos tanto de forma embarcada como a partir da Terra. Além disso, os desenvolvedores devem aderir a padrões rigorosos de segurança e qualidade, pois o fracasso de uma missão pode resultar em custos financeiros e humanos significativos.

O primeiro estágio em qualquer processo de design de software é desenvolver uma compreensão das relações entre o software que está sendo projetado e seu ambiente externo. Isto é essencial para decidir como fornecer a funcionalidade necessária do sistema e como estruturá-lo para se comunicar com seu ambiente. A compreensão do contexto também permite estabelecer os limites do sistema ajudar a decidir quais recursos serão implementados.

Para ajudar a monitorizar as alterações climáticas e melhorar a precisão das previsões meteorológicas em áreas remotas, o governo de um país com grandes áreas de floresta decide implantar várias centenas de estações meteorológicas nessas áreas remotas. Estas estações meteorológicas recolhem dados de um conjunto de instrumentos que medem temperatura e pressão, insolação, precipitação, velocidade do vento e direção do vento. As estações meteorológicas em áreas remotas fazem parte de um sistema maior que é um sistema de informação meteorológica que coleta dados de estações meteorológicas e os disponibiliza para processamento em outros sistemas.

Na Figura 1, foi usado o diagrama de pacotes UML para indicar que cada sistema é uma coleção de componentes, usando o estereótipo UML «system». As associações entre os pacotes indicam que há troca de informações.

Figura 1 – Ambiente da estação meteorológica



Os sistemas representados na Figura 1 são:

1. O sistema **Estação meteorológica**: é responsável por coletar dados meteorológicos, realizar processamento inicial de dados e transmiti-los ao sistema de gerenciamento de dados.
2. O sistema **Gerenciamento** (e Arquivamento) **de dados**: este sistema recolhe os dados de todas as estações meteorológicas remotas, realiza o processamento e análise de dados e arquiva os dados num formato que pode ser recuperado por outros sistemas, tais como sistemas de previsão meteorológica.
3. O sistema **Estação de manutenção** da estação: este sistema pode comunicar por satélite com todas as estações meteorológicas remotas para monitorizar a saúde destes sistemas e fornecer relatórios de problemas. Ele também pode atualizar o software embarcado nesses sistemas. No caso de problemas no sistema, ele também pode ser usado para controlar remotamente um sistema meteorológico remoto.

Cada estação meteorológica inclui uma série de instrumentos que medem parâmetros meteorológicos, como velocidade e direção do vento, temperaturas do solo e do ar, pressão barométrica e precipitação durante um período de 24 horas. Cada um desses instrumentos é controlado por um sistema de software que faz leituras periódicas dos parâmetros e gerência os dados coletados dos instrumentos.

O sistema de estação meteorológica opera coletando observações meteorológicas em intervalos frequentes – por exemplo, as temperaturas são medidas a cada minuto. No entanto, como a largura de banda do satélite é relativamente estreita, a estação meteorológica realiza algum processamento local e agregação dos dados. Em seguida, transmite (reporta) esses dados agregados quando solicitado pelo sistema de coleta de dados. Se, por qualquer motivo, for impossível estabelecer uma ligação, a estação meteorológica mantém os dados localmente até que a comunicação possa ser retomada.

Cada estação meteorológica é alimentada por bateria e deve ser totalmente independente – não há alimentação externa ou cabos de rede disponíveis. Todas as comunicações são feitas através de um link de satélite de velocidade relativamente lenta e a estação meteorológica deve incluir algum mecanismo (energia solar ou eólica) para carregar suas baterias. Como são implantados em áreas remotas, ficam expostos a condições ambientais severas e podem ser danificados por animais.

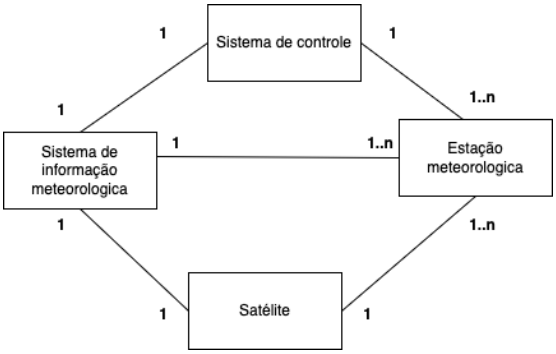
O software da estação, portanto, não se preocupa apenas com a coleta de dados e reporte dos dados e deve também:

1. Monitorar os instrumentos (reportar status), a energia e o hardware de comunicação e relatar as falhas ao sistema de gerenciamento.
2. Gerenciar a energia do sistema, garantindo que as baterias sejam carregadas sempre que as condições ambientais permitirem, mas também que os geradores sejam desligados em condições climáticas potencialmente prejudiciais, como ventos fortes.
3. Permitir a reconfiguração dinâmica onde partes do software são substituídas por novas versões e onde instrumentos de backup são transferidos para o sistema em caso de falha do sistema.

Como as estações meteorológicas têm de ser independentes e autônomas, isso significa que o software instalado é complexo, embora a funcionalidade de coleta de dados seja bastante simples.

A Figura 2 apresenta os sistemas no ambiente que se relaciona com cada Estação meteorológica: um Sistema de informação meteorológica, um sistema Satélite e um Sistema de controle. A informação de cardinalidade no link mostra que existe um sistema de controle, mas várias estações meteorológicas, um satélite e um sistema geral de informações meteorológicas.

Figura 2 – Contexto do ambiente de uma Estação Meteorológica.



A) Os casos de uso UML especificam o comportamento esperado (o quê), e não o método exato de fazer isso acontecer (como). Os casos de uso, uma vez especificados, podem ser denotados como representação textual e visual (diagrama de casos de uso). Usando o texto descrito neste enunciado que resultaram no diagrama de Contexto - Figura2, desenhe o modelo de casos de uso UML do Sistema de Informação Meteorológica e do Sistema de Controle.

B) Usando a notação gráfica UML para classes de objetos, projete as classes de objetos listadas a seguir, identificando atributos e operações. Use sua própria experiência para decidir sobre os atributos e operações que devem ser associados aos seguintes objetos:

1) Estação meteorológica

2) Dados meteorológicos

3) Termômetro de Solo

4) Anemômetro

5) Barômetro

C) Imagine agora que o sistema de Estação Meteorológico precisa decidir que tipo de barramento de comunicação deverá ser utilizado. Serial ou Paralelo? Utilizando o método de bit de paridade para detecção de erro?

C_1 Cite as vantagens e desvantagens de cada uma das comunicações: serial e da paralela, principalmente nos sistemas atuais.

C_2 Descreva como funciona o método de bit de paridade para detecção de erros de dados transmitidos.

D) A gerência do projeto da Estação Meteorológica solicitou também que se avaliasse melhor um método de tolerância a falhas para este sistema, tendo que decidir ente uma solução de *hardware*, *software* ou redundância híbrida. Comente pelo menos dois métodos de tolerância a falhas:

D_1 Método de tolerância a falhas de *hardware*.

D_2 Método de tolerância a falhas de *software*.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Questão 3

Surpreendentemente, após anos de espera, a cantora Madonna retornou ao país, num show inédito no Rio de Janeiro. A procura por ingressos se tornou absurda e filas gigantescas se formaram.

Com estas filas enormes, pode acontecer de a paciência de pessoas ir se esgotando e que elas acabem por desistir da compra de ingressos, liberando as filas e deixando ingressos para outras pessoas. Ao deixar a fila, a pessoa deixa vago o seu lugar e todas as pessoas que estão atrás dela dão um "passo a frente", sem nunca deixar vago o espaço entre duas pessoas.

Neste contexto, responda ao que se pede a seguir.

A) Identifique a melhor estrutura de dados para resolver o problema.

B) Explique por quê.

C) Apresente ainda as diferenças entre essa estrutura de dados em fila e a estrutura de dados em pilha.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Questão 4

Considere o seguinte programa desenvolvido na linguagem de programação C. O número que aparece ao lado de cada instrução a identifica.

```
1     #include <stdio.h>
2
3     void minha_funcao(unsigned char x1, unsigned char x2, int y) {
4           unsigned char r;
5           unsigned char rv[] = {'o', 'n', 'x', 'e'};
6           int i;
7           y = y < 50 ? 240 : 120;
8           r = x1 & y;
9           printf("Item 1: %d\n", r);
10          r = x2 >> 3;
11          printf("Item 2: %d\n", r);
12          rv[0] = x1, rv[2] = x2;
13          printf("Item 3: ");
14          for(i = 0; i < sizeof(rv); i++)
15                printf("%c", rv[i]);
16          if (y >= 60){
17                y = y - (6+12*2-8);
18          }
19          printf("\nItem 4: %d\n", y);
20    }
21
22    int main()
23    {
24          unsigned char a1 = 'I', a2 = 'p';
25          int b = 80;
26          minha_funcao(a1, a2, b);
27          return 0;
28    }
```

Para a resolução dessa questão, considere que: (i) o caractere “I” (i maiúsculo) é representado pelo número 73 no sistema de numeração decimal, o que equivale ao número 0x49 no sistema de numeração hexadecimal (0x representa que o número à frente está no sistema hexadecimal); (ii) O caractere “p” (p minúsculo) é representado pelo número 112 no sistema decimal, o que equivale a 0x70 no sistema hexadecimal; (iii) O número 80 no sistema decimal equivale a 0x50 no sistema hexadecimal; (iv) O número 120 no sistema decimal equivale a 0x78 no sistema hexadecimal; (v) O número 240 no sistema decimal equivale a 0xF0 no sistema hexadecimal.

O programa mostra, ao ser executado, 4 linhas como saída, devido a chamadas a função `printf` no contexto da função `minha_funcao` (vide “Item 1”, “Item 2”, “Item 3” e “Item 4” relacionados à função `printf`). Com base nesses dados, responda ao que se pede a seguir.

A) Qual é a saída (linha completa mostrada na saída devido a execução do programa) relacionada ao “Item 1”? Explique detalhadamente sua resposta.

B) Qual é a saída (linha completa mostrada na saída devido a execução do programa) relacionada ao “Item 2”? Explique detalhadamente sua resposta.

C) Qual é a saída (linha completa mostrada na saída devido a execução do programa) relacionada ao “Item 3”? Explique detalhadamente sua resposta.

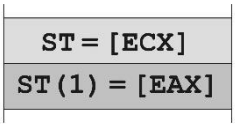
D) Qual é a saída (linha completa mostrada na saída devido a execução do programa) relacionada ao “Item 4”? Explique detalhadamente sua resposta.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Questão 5

A Unidade de Ponto Flutuante (FPU) tradicional de um processador x86 CISC trabalha com uma pilha interna de até 8 níveis para realizar operações com reais de 64 bits (precisão dupla). Ela permite a operação direta com dois operandos empilhados. Por exemplo, seja a figura abaixo com os dados empilhados representados por **ST** e **ST(1)**, sendo que **ST(1)** foi empilhado primeiro e **ST** foi empilhado por último, ficando no topo da pilha, via código em assembly a seguir.

```
;empilha real de precisão dupla apontado por EAX
fld qword ptr[eax]     ;ST ← [EAX] (push)
;empilha real de precisão dupla apontado por ECX
fld qword ptr[ecx]     ;ST(1) ← ST e depois ST ← [ECX] (push)
```



Como exemplo, para realizar uma soma entre os dois últimos dados empilhados, usa-se a instrução **faddp st(1),st**. Como nas instruções convencionais com inteiros, armazena-se o resultado no destino (operando da esquerda). A principal particularidade desta instrução é que, após a operação ser realizada, o topo da pilha (**ST**) é descartado *(pop)* e o resultado armazenado em **ST(1)** passa a ser o novo topo da pilha (**ST**).

```
faddp st(1),st      ;ST(1) ← ST(1)+ST, pop (ST) e ST ← ST(1)
```

Para salvar na memória (no endereço apontado pelo registrador EDX) o resultado da soma e descartá-lo da pilha *(pop)*, basta executar a instrução **fstp**.

```
fstp qword ptr[edx]     ;[EDX] ← ST e pop topo da pilha (ST)
```

Sejam algumas instruções do assembly para operações usando a FPU.

f2xmi – faz ST ← 2^{ST}-1.
faddp – faz destino ← destino + fonte e faz *pop* da pilha.
fchs – faz ST ← -ST.
fcos – faz ST ← cos(ST).
fld – faz *push* da pilha e ST ← fonte.
fldpi – faz *push* de π na pilha e ST ← π.
fld1 – faz *push* de +1.0 na pilha e ST ← +1.0.
fmulp – faz destino ← destino x fonte e faz *pop* da pilha.
fstp – faz destino ← ST e faz *pop* da pilha.
fyl2x – faz ST(1) ← ST(1) x $\log_2$ST e faz *pop* da pilha.

Diante do exposto:

A) escreva um código em assembly apenas com as instruções listadas para realizar, em precisão dupla, a operação: [EAX] ← cos (π [EAX] $^{-[ECX]}$)

B) Após a execução de cada instrução do código escrito no item anterior, indique a configuração da pilha com os dados nela inseridos e seus respectivos valores.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Realização