RESPOSTAS

1-) 🡪 Um exemplo de software embarcado é o sistema operacional Android, desenvolvido pela Google, que é utilizado em diversos dispositivos móveis, como smartphones e tablets. O Android é responsável por gerenciar os recursos do dispositivo, como a tela, processador, memória, rede e sensores, além de fornecer uma interface gráfica para o usuário interagir com o dispositivo. As características de software embarcados incluem o fato de que eles são projetados para executar em hardware específico, muitas vezes com recursos limitados, e são geralmente desenvolvidos em linguagens de programação de baixo nível, como C e Assembly. Eles também são projetados para serem confiáveis, seguros e eficientes em termos de energia. Empresas que desenvolvem software embarcado incluem a Intel, a Qualcomm, a Texas Instruments e a ARM, entre outras. Além disso, muitas empresas de hardware, como fabricantes de automóveis e de dispositivos médicos, também desenvolvem software embarcado para seus produtos.

2-) 🡪 A área de Engenharia de Software é mega importante para a computação porque os softwares estão cada vez mais complicados e críticos para muitas atividades, e a Engenharia de Software tem um monte de práticas e métodos para lidar com essa complicação e garantir a qualidade dos softwares feitos. A Engenharia de Software também ajuda a fazer programas de computador de um jeito organizado e sistemático, o que é fundamental para garantir que o software seja confiável e fácil de manter no longo prazo. Com a Engenharia de Software, é possível definir requisitos claros para o software, planejar seu desenvolvimento, monitorar sua qualidade e testá-lo direitinho. Outra coisa importante da Engenharia de Software é que ela ajuda a trabalhar junto com outras áreas do conhecimento, como Matemática, Eletrônica e Psicologia, por exemplo. A Engenharia de Software ajuda a juntar várias habilidades diferentes e fazer um processo de desenvolvimento de software que funcione bem. Por fim, a Engenharia de Software também ajuda a atender às necessidades cada vez maiores de softwares que sejam fáceis de usar, seguros e acessíveis para todo mundo. A Engenharia de Software permite que os desenvolvedores criem soluções que atendam às necessidades dos usuários, que possam crescer conforme a necessidade e que possam ser mantidas e atualizadas de forma eficiente.

3-) 🡪 A Engenharia de Software se tornou necessária nos anos 60 devido à complexidade crescente dos softwares e à demanda por programas de alta qualidade em grande escala. Hoje em dia, os desafios da Engenharia de Software continuam a existir, mas em uma escala ainda maior, devido ao aumento da interconexão e da complexidade dos softwares. A Engenharia de Software é fundamental para garantir a confiabilidade, escalabilidade e eficiência dos softwares e os desenvolvedores precisam utilizar as melhores práticas e metodologias disponíveis para lidar com esses desafios.

4-) 🡪  O objetivo da conferência era chamar a atenção para um problema crucial do uso de computadores, o chamado software. A conferência produziu um relatório, com mais de 130 páginas, que afirmava a necessidade de que software fosse construído com base em princípios práticos e teóricos, tal como ocorre em ramos tradicionais e bem estabelecidos da Engenharia. Para deixar essa proposta mais clara, decidiu-se cunhar o termo Engenharia de Software. Por isso, a Conferência da OTAN é considerada o marco histórico de criação da área de Engenharia de Software.

5-) 🡪 Essa afirmação é baseada na ideia de que o desenvolvimento de software é uma atividade criativa e imprevisível, que não pode ser totalmente planejada e projetada de antemão. Diferentemente de produtos tradicionais de engenharia, como pontes e edifícios, o software é flexível e pode ser alterado facilmente durante o processo de desenvolvimento. Por isso, as metodologias de desenvolvimento de software modernas enfatizam a colaboração, a comunicação e a adaptação às mudanças durante todo o processo, em vez de uma abordagem estritamente sequencial. Essa abordagem permite que os desenvolvedores trabalhem de forma mais eficiente e efetiva, produzindo software de alta qualidade que atenda às necessidades dos usuários.

6-) 🡪 Não há "balas de prata" na área de TI, ou seja, não existe uma solução única e infalível que resolva todos os problemas de software. Cada projeto de TI é único e apresenta desafios próprios, e a solução efetiva depende de uma combinação de fatores, como escolha adequada de tecnologias, processos de desenvolvimento eficientes, colaboração entre equipe e clientes, entre outros.

7-) 🡪**Complexidade**: dentre as construções que o homem se propõe a realizar, software é uma das mais desafiadoras e mais complexas que existe. Na verdade, como dissemos antes, mesmo construções de engenharia tradicional, como um satélite, uma usina nuclear ou um foguete, são cada vez mais dependentes de software.  
**Conformidade**: pela sua natureza software tem que se adaptar ao seu ambiente, que muda a todo momento no mundo moderno. Por exemplo, se as leis para recolhimento de impostos mudam, normalmente espera-se que os sistemas sejam rapidamente adaptados à nova legislação. Brooks comenta que isso não ocorre, por exemplo, na Física, pois as leis da natureza não mudam de acordo com os caprichos dos homens.  
**Facilidade de mudanças**: que consiste na necessidade de evoluir sempre, incorporando novas funcionalidades. Na verdade, quanto mais bem sucedido for um sistema de software, mais demanda por mudanças ele recebe.  
**Invisibilidade**: devido à sua natureza abstrata, é difícil visualizar o tamanho e consequentemente estimar o esforço de construir um sistema de software.

8-) 🡪 **Requisitos funcionais:**  
Sistema de cadastro de clientes, permitindo que os usuários se cadastrem e criem uma conta na loja.

Sistema de busca, permitindo que os usuários procurem por produtos específicos na loja.

Sistema de carrinho de compras, permitindo que os usuários adicionem produtos ao carrinho e efetuem a compra.

**Requisitos não funcionais:**

Segurança, garantindo que as informações do cliente e do cartão de crédito sejam protegidas.

Performance, garantindo que o site carregue rapidamente e responda de forma ágil.

Usabilidade, garantindo que o site seja intuitivo e fácil de usar, com design atraente e responsivo.

9-) 🡪 A área de projeto de software tem como objetivo definir a estrutura e a arquitetura do software, incluindo a seleção das classes e componentes que serão utilizados no sistema e como eles se relacionam entre si. Isso implica em encapsular dados e comportamentos em objetos, criar interfaces e APIs para a comunicação entre módulos e garantir que o software seja modular, flexível e escalável. Além disso, o projeto pode utilizar padrões de design para solucionar problemas comuns de desenvolvimento de software e assegurar a qualidade e manutenibilidade do código.

10-) 🡪 Não necessariamente. A existência da classe ContaBancaria não depende diretamente da existência da classe Cliente, mas a sua funcionalidade sim. A dependência entre as classes significa que a ContaBancaria precisa da interface fornecida pela classe Cliente para realizar operações como verificar o saldo disponível ou realizar transferências.

No contexto da loja de calçados, uma classe com uma interface requerida poderia ser a classe de Pagamento, que depende da classe de Cartão de Crédito para processar as transações de compra. Ou seja, a classe de Pagamento possui uma dependência para a classe de Cartão de Crédito, já que precisa da sua interface para realizar as operações necessárias para concluir a transação.

11-) 🡪 Construção trata da implementação, isto é, codificação do sistema. Nesse momento, existem diversas decisões que precisam ser tomadas, como, por exemplo: definir os algoritmos e estruturas de dados que serão usados, definir os frameworks e bibliotecas de terceiros que serão usados; definir técnicas para tratamento de exceções; definir padrões de nomes, leiaute e documentação de código e, por último, mas não menos importante, definir as ferramentas que serão usadas no desenvolvimento, incluindo compiladores, ambientes integrados de desenvolvimento (IDEs), depuradores, gerenciadores de bancos de dados, ferramentas para construção de interfaces, etc.

12-) 🡪 **Testes de unidade:** testam a funcionalidade de cada unidade do sistema de forma isolada, como as classes de Produto, CarrinhoDeCompras e Pagamento.

**Testes de integração:** testam a integração entre diferentes unidades do sistema, como a conexão entre a classe de Produto e a de CarrinhoDeCompras.

**Testes de sistema:** testam o sistema como um todo, incluindo a interface do usuário, fluxos de navegação e integração com outros sistemas externos.

**Testes de aceitação:** testam se o sistema atende aos requisitos especificados pelo cliente, como a capacidade de adicionar produtos ao carrinho de compras e concluir uma transação de pagamento com sucesso.

**Testes de desempenho:** testam a capacidade do sistema de lidar com uma carga de trabalho maior do que o usual, como o número de usuários acessando o site simultaneamente.

**Testes de segurança:** testam a segurança do sistema contra ataques externos, como tentativas de invasão e roubo de dados dos usuários.

13-) 🡪 **Exemplos de manutenções:** correção de bugs, ajustes de performance, atualização de dependências.  
**Exemplos de evoluções:** adição de novas funcionalidades, integração com novos sistemas de pagamento, melhorias na interface do usuário.

14-) 🡪 Exemplos de stakeholders: dono da loja de calçados, gerente de vendas, equipe de desenvolvimento, equipe de testes, usuários finais do sistema.

16-) 🡪 a-) O processo Waterfall é um modelo de desenvolvimento sequencial, onde as atividades são executadas em fases sequenciais, seguindo uma ordem fixa e rígida, desde a análise de requisitos até a manutenção do sistema.

b-) O processo ágil é um modelo de desenvolvimento iterativo e incremental, onde o desenvolvimento é realizado em ciclos curtos e repetitivos chamados de iterações, focando na entrega contínua de valor ao cliente e na adaptação às mudanças durante o processo de desenvolvimento.

17-) 🡪 Não há um método único ou mais utilizado nas empresas de desenvolvimento, pois cada organização pode escolher e adaptar um método que melhor se adapte às suas necessidades e objetivos. No entanto, o Scrum é um dos métodos ágeis mais populares e amplamente utilizados.

1. Metodologia híbrida é uma abordagem de desenvolvimento de software que combina práticas e técnicas de diferentes metodologias de desenvolvimento de software, como Agile e Waterfall, para atender às necessidades específicas de um projeto. A metodologia híbrida é usada para aproveitar as vantagens de diferentes metodologias e minimizar suas desvantagens.

18-) 🡪 Modelos de software são abstrações que representam diferentes visões do sistema a ser desenvolvido, permitindo a análise, design e implementação do software. Eles servem para guiar o processo de desenvolvimento de software, definindo fases e atividades necessárias para alcançar os objetivos do projeto. Modelos de software são utilizados em projetos de desenvolvimento de software, permitindo uma abordagem sistemática e organizada para alcançar resultados consistentes. Exemplos incluem o modelo em cascata, modelo iterativo/incremental, modelo V, modelo espiral, entre outros.

19-) 🡪 A engenharia avante é usada quando se deseja construir um novo sistema de software, enquanto a engenharia reversa é usada quando se deseja entender ou modificar um sistema de software existente. Ambas as técnicas são usadas em diferentes momentos do ciclo de vida do software.

20-) 🡪 Os usuários que poderiam avaliar a qualidade externa do software da loja de calçados incluem clientes, revisores de software, especialistas em usabilidade, especialistas em experiência do usuário, entre outros.

21-) 🡪 As qualidades externas mais preponderantes em um software são:

* Usabilidade: porque um software fácil e intuitivo de usar pode garantir uma boa experiência do usuário e reduzir a probabilidade de erros e retrabalho.
* Eficiência: porque um software que responde rapidamente às ações do usuário ou processa grandes quantidades de dados em um tempo razoável pode garantir uma maior satisfação do usuário e aumentar a produtividade.
* Confiabilidade: porque um software que é capaz de operar continuamente sem falhas ou erros pode garantir a segurança e a integridade dos dados do usuário e evitar perdas financeiras ou danos à reputação da empresa.

22-) 🡪As qualidades internas de software são relacionadas à sua estrutura e características internas. Algumas das principais qualidades internas incluem: modularidade, coesão, acoplamento, escalabilidade, testabilidade, manutenibilidade, entre outras. A modularidade refere-se à capacidade do software de ser dividido em módulos independentes e interconectados. A coesão está relacionada à capacidade dos módulos de executar uma única função. O acoplamento refere-se à dependência entre módulos. A escalabilidade está relacionada à capacidade do software de lidar com um grande volume de dados. A testabilidade é a capacidade do software de ser testado de forma eficiente. A manutenibilidade refere-se à facilidade com que o software pode ser modificado e mantido ao longo do tempo.

23-) 🡪 O Código de Ética da Sociedade Brasileira de Computação (SBC) é um conjunto de princípios éticos que orientam a conduta de profissionais da área de tecnologia da informação no Brasil. O código estabelece diretrizes que buscam garantir que o desenvolvimento, uso e disseminação de tecnologia sejam realizados de forma ética e responsável. Em minha percepção, o Código de Ética da SBC é uma ferramenta importante para a conscientização dos profissionais da área de TI sobre a importância da ética e responsabilidade na sua atuação. O cumprimento dos princípios estabelecidos pelo código é fundamental para a construção de uma sociedade mais justa e igualitária, em que a tecnologia seja utilizada em benefício do bem comum.

1. 24-) 🡪   
   1 . Software de sistema: são programas que gerenciam recursos de hardware e software do computador, fornecendo serviços comuns a outros programas. Exemplos incluem sistemas operacionais, drivers de dispositivos e utilitários do sistema.
2. Software de aplicativo: são programas projetados para executar tarefas específicas para os usuários, como processamento de texto, planilhas, jogos, navegadores da web e sistemas de gerenciamento de bancos de dados.
3. Software de programação: são ferramentas usadas pelos desenvolvedores de software para criar outros programas. Exemplos incluem compiladores, editores de código, depuradores e ambientes de desenvolvimento integrado (IDEs).

As qualidades internas de software referem-se às características que tornam o software eficiente, legível, fácil de modificar e manter. Isso inclui, entre outras coisas:

* Modularidade: o software é dividido em módulos independentes, o que torna mais fácil alterar ou adicionar funcionalidades sem afetar o resto do sistema.
* Coesão: módulos são projetados para executar uma única tarefa bem definida.
* Acoplamento: módulos são projetados para serem interdependentes apenas na medida necessária para realizar suas funções.
* Legibilidade: o código-fonte é fácil de ler e entender.
* Manutenibilidade: o software é fácil de manter e corrigir problemas.
* Portabilidade: o software pode ser executado em diferentes plataformas sem muitas alterações no código.