Dal problema al programma

Compito di realtà: **Creare un progetto software che risolva un problema reale.** Gli studenti devono identificare un problema nella loro scuola o comunità, progettare un programma per risolverlo e implementarlo.

Ad esempio, un'applicazione per la gestione delle prenotazioni della biblioteca scolastica o un sistema per monitorare il consumo energetico della scuola.

Soluzione dei problemi: processi euristici e processi algoritmici

Compito di realtà: Gli studenti devono scegliere un problema complesso (ad esempio, il percorso più breve per visitare tutti i punti di interesse della propria città) e confrontare soluzioni basate su algoritmi esatti (come l'algoritmo di Dijkstra) e soluzioni euristiche

(banalmente, provare tutte le strade percorribili). Necessario documentare i pro e i contro di ciascun approccio e presentare i risultati in una relazione scritto, eventualmente da discutere in classe con moderazione del docente.

Tecniche di rappresentazione degli algoritmi: flow chart, pseudocodice

Compito di realtà: Progettare un diagramma di flusso e scrivere il pseudo-codice per un sistema di gestione delle presenze scolastiche in classe. Gli studenti devono prima raccogliere i requisiti del sistema, quindi creare un diagramma di flusso dettagliato e uno pseudo-codice che rappresenti il funzionamento del sistema.

Proprietà degli algoritmi: la programmazione strutturata, complessità

Compito di realtà:

1. **Analisi dell'algoritmo di ricerca lineare**: Gli studenti devono selezionare un algoritmo di ricerca semplice, come la ricerca lineare in un array non ordinato. Devono implementare l'algoritmo e analizzarne la complessità temporale e spaziale. Devono documentare il processo e presentare i risultati.
2. **Miglioramento dell'algoritmo**: Gli studenti devono esplorare modi per migliorare l'efficienza dell'algoritmo di ricerca. Ad esempio, possono implementare un algoritmo di ricerca binaria per array ordinati, che ha una complessità temporale migliore. Devono confrontare le prestazioni dei due algoritmi e analizzare i risultati.
3. **Documentazione del processo di miglioramento**: Gli studenti devono scrivere un rapporto che descriva il processo di miglioramento dell'algoritmo di ricerca. Devono spiegare le differenze tra i due algoritmi, analizzare i vantaggi e gli svantaggi di ciascuno e riflettere sulle lezioni apprese durante il processo.

Questo compito offre agli studenti l'opportunità di applicare concetti di complessità algoritmica in modo più accessibile, utilizzando algoritmi di ricerca semplici. Inoltre, promuove lo sviluppo delle capacità di analisi critica, risoluzione dei problemi e documentazione.

Algoritmi notevoli: (es. ordinamento, ricerca, fusione)

Compito di realtà: Implementare e confrontare diversi algoritmi di ordinamento.

1. **Selezione degli algoritmi di ordinamento**: Gli studenti devono selezionare due algoritmi di ordinamento comuni e relativamente semplici, come il Bubble Sort e l'Insertion Sort, che possono essere compresi e implementati facilmente.
2. **Implementazione degli algoritmi**: Gli studenti devono implementare entrambi gli algoritmi di ordinamento in un linguaggio di programmazione di loro scelta. Possono utilizzare risorse online, tutorial e guide per aiutarli nell'implementazione.
3. **Confronto delle prestazioni**: Una volta implementati, gli studenti devono confrontare le prestazioni dei due algoritmi utilizzando un set di dati di

dimensioni crescenti. Possono misurare il tempo necessario per ordinare gli array di diversa dimensione e confrontare i risultati.
4. **Analisi dei risultati**: Gli studenti devono analizzare e confrontare i tempi di esecuzione ottenuti dai due algoritmi. Possono considerare fattori come la complessità temporale, il comportamento su dati ordinati e non ordinati e la facilità di implementazione.
5. **Rapporto finale**: Gli studenti devono scrivere un rapporto che descriva il processo di implementazione e confronto degli algoritmi di ordinamento. Devono spiegare le differenze tra i due algoritmi, analizzare i risultati del confronto delle prestazioni e trarre conclusioni sulle loro osservazioni.

Linguaggi formali: sintassi e semantica

Compito di realtà: Progettare un semplice linguaggio di programmazione (sulla falsariga di LOGO o Turtle di Python).

1. **Definizione del linguaggio**: Gli studenti devono definire la sintassi e la semantica di un linguaggio di programmazione minimalista progettato specificamente per la creazione di giochi semplici. Il linguaggio dovrebbe includere comandi per disegnare forme grafiche di base (come cerchi, rettangoli, linee), gestire l'input utente (tasti premuti, movimento del mouse) e implementare semplici strutture di controllo (come cicli e condizionali).
2. **Implementazione di un interprete**: Gli studenti devono scrivere un interprete per questo linguaggio di programmazione utilizzando un linguaggio di programmazione di loro scelta. L'interprete dovrebbe essere in grado di leggere e eseguire i programmi scritti nel linguaggio definito dagli studenti.
3. **Dimostrazione del funzionamento**: Gli studenti devono dimostrare il funzionamento del linguaggio di programmazione e dell'interprete attraverso la creazione di giochi semplici. Possono creare giochi come Pong, Snake, Tic Tac Toe o altri giochi a loro scelta utilizzando il linguaggio e l'interprete che hanno creato.
4. **Riflessione e documentazione**: Gli studenti devono scrivere una breve riflessione che documenti il processo di progettazione del linguaggio, l'implementazione dell'interprete e la creazione dei giochi. Possono anche riflettere sull'esperienza di programmazione e sui concetti imparati durante il processo.

Intelligenza artificiale: problem solving, ragionamento, rappresentazione della conoscenza, apprendimento automatico

Compito di realtà: Creare un sistema di intelligenza artificiale per risolvere un problema specifico.

Gli studenti devono scegliere un'area di applicazione (ad esempio, un *chatbot* per rispondere a domande frequenti degli studenti) e sviluppare un prototipo funzionante utilizzando semplici tecniche di AI.

Sviluppo di un chatbot per rispondere a domande frequenti degli studenti

1. **Scelta dell'area di applicazione**: Gli studenti devono scegliere un'area specifica in cui il chatbot sarà utile. Ad esempio, potrebbe essere un chatbot per una

scuola che risponde alle domande frequenti degli studenti su argomenti come gli orari delle lezioni, le date degli esami, le attività extracurricolari, ecc.
2. **Sviluppo del prototipo**: Gli studenti devono utilizzare Python per sviluppare un prototipo funzionante del chatbot. Possono utilizzare librerie come NLTK (Natural Language Toolkit) o spaCy per elaborare il linguaggio naturale e rispondere alle domande degli studenti in modo intelligente.
3. **Implementazione delle tecniche di AI**: Gli studenti devono implementare semplici tecniche di intelligenza artificiale per migliorare le capacità del chatbot nel comprendere e rispondere alle domande degli studenti. Ad esempio, possono utilizzare algoritmi di ricerca delle parole chiave o classificatori per identificare il tipo di domanda e generare una risposta appropriata.
4. **Test e valutazione**: Gli studenti devono testare il prototipo del chatbot utilizzando una serie di domande simulate dagli studenti. Possono valutare le prestazioni del chatbot in base alla sua capacità di comprendere le domande e fornire risposte accurate e utili.
5. **Riflessione e miglioramento**: Gli studenti devono riflettere sul processo di sviluppo del chatbot e identificare eventuali aree di miglioramento. Possono considerare nuove funzionalità da aggiungere, modi per ottimizzare le prestazioni del chatbot e come rendere l'esperienza degli utenti più intuitiva e soddisfacente.

Devono documentare il processo di sviluppo, inclusi i metodi di rappresentazione della conoscenza, il ragionamento e l'apprendimento automatico utilizzati.

Codifica digitale delle informazioni

Compito di realtà: Realizzare un progetto che esplori i metodi di codifica delle informazioni digitali.

Spiegare e sperimentare con metodi di codifica informatica

1. **Scelta dei metodi di codifica**: Gli studenti devono selezionare diversi metodi di codifica comuni, come binario, esadecimale, ASCII e UTF-8, da includere nella loro presentazione.
2. **Spiegazione dei metodi di codifica**: Gli studenti devono spiegare in modo chiaro e semplice ciascun metodo di codifica, descrivendo come funziona e dove viene utilizzato nei computer. Ad esempio, possono spiegare che il sistema binario utilizza solo due cifre (0 e 1) per rappresentare i dati, mentre l'ASCII e l'UTF-8 sono standard utilizzati per rappresentare caratteri testuali.
3. **Illustrazione con esempi pratici**: Gli studenti devono includere esempi pratici per dimostrare come funzionano i diversi metodi di codifica. Possono mostrare come un carattere o un numero viene rappresentato in binario, esadecimale, ASCII e UTF-8, e confrontare le differenze tra di loro.
4. **Attività interattive**: Gli studenti possono coinvolgere i loro compagni di classe in attività pratiche, come la codifica e la decodifica di messaggi utilizzando i diversi metodi di codifica. Possono creare semplici esercizi in cui gli studenti devono convertire un messaggio da un formato all'altro e verificare i risultati.
5. **Riflessione e discussione**: Alla fine della presentazione, gli studenti devono incoraggiare una discussione sulle applicazioni pratiche dei metodi di codifica nei computer e sulle sfide e le considerazioni legate alla scelta del metodo di codifica più appropriato per una determinata situazione.

Questo compito offre agli studenti l'opportunità di apprendere e sperimentare con concetti di codifica informatica in modo pratico e coinvolgente. La presentazione e le attività interattive consentono loro di comprendere meglio come i computer elaborano e memorizzano le informazioni utilizzando diversi metodi di codifica.

Sistemi digitali e programmabili: i microprocessori, programmazione a livello macchina e con linguaggi orientati alla macchina

Compito di realtà: **Scrivere e eseguire un programma in linguaggio assembly su un microprocessore simulato**

1. **Apprendimento delle basi della programmazione assembly**: Gli studenti devono imparare i concetti di base della programmazione assembly, come l'utilizzo dei registri, le istruzioni di caricamento e memorizzazione dei dati e le operazioni aritmetiche.
2. **Sviluppo di un programma semplice**: Gli studenti devono scrivere un programma assembly utilizzando un emulatore di CPU come SPIM per MIPS. Il programma deve eseguire un'operazione aritmetica di base, come l'addizione o la sottrazione, su due numeri e visualizzare il risultato.
3. **Esecuzione del programma e documentazione**: Gli studenti devono eseguire il programma sull'emulatore di CPU e verificare che produca il risultato desiderato. Devono quindi documentare il codice assembly e spiegare il funzionamento del programma, illustrando passo dopo passo cosa fa ciascuna istruzione.

Questo compito offre agli studenti un'introduzione pratica alla programmazione assembly e all'utilizzo di microprocessori simulati. La semplicità del programma richiesto e l'utilizzo di un emulatore di CPU semplificano il processo di apprendimento e consentono agli studenti di concentrarsi sui concetti fondamentali della programmazione a basso livello.

Componenti di un sistema di elaborazione: motherboard, unità centrale, unità periferiche, memorie e loro caratteristiche e gerarchia (Von Neumann)

Compito di realtà: Smontare e rimontare un computer in laboratorio.

Gli studenti devono identificare e descrivere ciascun componente della motherboard, l'unità centrale, le periferiche e le varie memorie. Devono poi creare una guida illustrata che spieghi il ruolo di ogni componente e come si interconnettono secondo l'architettura di Von Neumann.

Elaboratori monoprocessore: tipologie di architetture e loro caratteristiche funzionali

Compito di realtà: Ricercare e presentare un rapporto sulle diverse tipologie di architetture di elaboratori monoprocessore (ad esempio, CISC vs. RISC). Gli studenti devono spiegare le caratteristiche funzionali di ciascun tipo, i vantaggi e gli svantaggi, e fornire esempi di processori reali che utilizzano queste architetture.

Architetture parallele. Sistemi multiprocessori superscalari, sistemi a memoria condivisa, sistemi a memoria distribuita, sistemi a matrice

Compito di realtà: Creare una presentazione che riassuma i diversi tipi di architetture parallele. Gli studenti devono includere spiegazioni dettagliate di sistemi multiprocessori superscalari, sistemi a memoria condivisa e distribuita, e sistemi a matrice. Devono usare diagrammi e animazioni per illustrare come funzionano queste architetture e dove vengono utilizzate.

1. **Introduzione alle architetture parallele**: Gli studenti devono spiegare il concetto di architetture parallele, evidenziando l'importanza di utilizzare più processori o core per eseguire operazioni simultaneamente e aumentare le prestazioni dei sistemi informatici.
2. **Sistemi multiprocessori superscalari**: Gli studenti devono descrivere i sistemi multiprocessori superscalari, che utilizzano più unità di esecuzione per eseguire istruzioni in parallelo. Possono includere esempi di marche come Intel con la loro serie di processori Xeon.
3. **Sistemi a memoria condivisa**: Gli studenti devono spiegare i sistemi a memoria condivisa, dove più processori accedono alla stessa memoria principale. Possono citare esempi come i server multi-core AMD Ryzen, che utilizzano questa architettura per migliorare le prestazioni.
4. **Sistemi a memoria distribuita**: Gli studenti devono descrivere i sistemi a memoria distribuita, in cui ogni processore ha la propria memoria locale e comunica con gli altri tramite una rete. Possono menzionare esempi come i cluster di server utilizzati nei data center delle grandi aziende come Google e Amazon.
5. **Sistemi a matrice**: Gli studenti devono spiegare i sistemi a matrice, che utilizzano una rete di processori collegati tra loro in una struttura a griglia. Possono fare riferimento a esempi di marche come NVIDIA con le loro GPU (Graphics Processing Unit) utilizzate per l'elaborazione parallela in applicazioni di intelligenza artificiale e calcolo scientifico.

Architettura dei microcontrollori e loro programmazione

Compito di realtà: **Progettare e programmare un dispositivo utilizzando Arduino**

1. **Selezione del progetto**: Gli studenti devono scegliere un progetto pratico da realizzare con Arduino, come un sistema di controllo delle luci, un sensore di temperatura, un allarme o un contatore. Possono basare la scelta del progetto sui loro interessi personali o sulle esigenze della comunità locale.
2. **Progettazione del circuito**: Gli studenti devono progettare il circuito utilizzando Arduino e i componenti elettronici necessari per il progetto scelto. Possono utilizzare software di progettazione come Fritzing per disegnare lo schema del circuito e pianificare la disposizione dei componenti.
3. **Assemblaggio del circuito**: Una volta progettato il circuito, gli studenti devono assemblare i componenti seguendo lo schema elettrico. Possono utilizzare una breadboard per facilitare l'assemblaggio e la prototipazione del circuito.
4. **Scrittura del codice**: Gli studenti devono scrivere il codice necessario per far funzionare il dispositivo Arduino. Possono utilizzare l'IDE di Arduino e il linguaggio di programmazione basato su C/C++ per sviluppare il software. Il codice dovrebbe

includere istruzioni per leggere i sensori, elaborare i dati e controllare i dispositivi di output.
5. **Test e debugging**: Una volta completata la programmazione, gli studenti devono testare il dispositivo per assicurarsi che funzioni correttamente. Possono eseguire una serie di test per verificare il corretto funzionamento del circuito e del software e risolvere eventuali problemi di funzionamento.
6. **Documentazione e presentazione**: Gli studenti devono documentare l'intero processo di progettazione e programmazione, dalla fase di progettazione del circuito alla scrittura del codice e ai test di funzionamento. Possono includere foto del circuito, lo schema elettrico, il codice sorgente e una descrizione dettagliata del funzionamento del dispositivo. Infine, devono presentare il prototipo funzionante ai compagni di classe o ad altri interessati.

Sistemi operativi: tipologie, architettura e funzioni

Compito di realtà: Creare un confronto tra diversi sistemi operativi (ad esempio, Windows, macOS, Linux). Gli studenti devono ricercare e documentare le caratteristiche principali di ciascun sistema operativo, inclusi la loro architettura, le funzioni, i vantaggi e gli svantaggi. Devono poi presentare i risultati in un formato visivo come presentazione o infografica / cartellone.

La gestione delle risorse fisiche e dei programmi da parte del sistema operativo. Analisi delle prestazioni

Compito di realtà: Monitorare e analizzare le prestazioni di un computer utilizzando strumenti di sistema operativi (ad esempio, Task Manager su Windows, Activity Monitor su macOS o htop su Linux). Gli studenti devono eseguire varie attività (come l'esecuzione di software intensivo di risorse, gestione di file di grandi dimensioni) e raccogliere dati sulle prestazioni del sistema (utilizzo della CPU, memoria, disco). Devono poi redigere un rapporto che analizza come il sistema operativo gestisce le risorse durante queste attività.

Problemi di parallelismo e concorrenza

Compito di realtà: Scrivere un programma semplice in un linguaggio di programmazione che supporti il multi-threading (ad esempio, Python, Java) per dimostrare la gestione della concorrenza. Gli studenti devono creare un programma che esegua operazioni parallele (ad esempio, calcolare la somma di grandi matrici) e documentare come il linguaggio e il sistema operativo gestiscono il parallelismo. Devono poi spiegare i problemi comuni di concorrenza, come race conditions e deadlocks, e come possono essere risolti.

Programmi di elaborazione dei linguaggi: interpreti e compilatori

Compito di realtà: Costruire un semplice interprete o compilatore per un linguaggio di programmazione minimalista. Gli studenti devono creare un linguaggio con sintassi e semantica di base, quindi scrivere un interprete o un compilatore che possa eseguire programmi scritti in quel linguaggio. Devono documentare il processo di sviluppo e spiegare le differenze tra interpreti e compilatori.

Software di utilità e software applicativi: software per l'automazione d'ufficio (SOHO)

Compito di realtà: Progettare e implementare una suite di strumenti di automazione d'ufficio utilizzando software disponibili (ad esempio, Google Workspace, Microsoft Office, LibreOffice). Gli studenti devono creare vari documenti, fogli di calcolo e presentazioni che automatizzano compiti comuni in un ufficio (ad esempio, un modello di bilancio, una lista di contatti automatizzata, una presentazione per riunioni). Devono poi presentare come queste automazioni migliorano l'efficienza e la produttività in un ambiente d'ufficio.

Fondamenti di comunicazioni: segnali, canali, mezzi e metodi di trasmissione (analisi funzionale)

Compito di realtà: Progettare un esperimento per dimostrare i principi di base della trasmissione dei segnali. Gli studenti devono utilizzare strumenti come oscilloscopi e generatori di segnali per mostrare come i segnali elettrici, ottici o radio vengono trasmessi attraverso diversi mezzi (cavi coassiali, fibre ottiche, onde radio). Devono poi documentare i risultati, spiegando i concetti di attenuazione, rumore e larghezza di banda.

Il modello ISO-OSI: livelli e primitive di interfaccia

Compito di realtà: Creare un gioco di ruolo in cui ogni studente rappresenta un livello diverso del modello ISO-OSI. Devono simulare la trasmissione di dati da un computer all'altro, passando le informazioni attraverso i vari livelli. Ogni studente deve spiegare le funzioni e le primitive di interfaccia del livello che rappresenta. Il gioco deve includere esempi pratici di come i dati vengono incapsulati e decapsulati.

La suite di protocolli TCP/IPv4: algoritmi di switching layer 2 e di routing layer 3. Protocollo IPv6

Compito di realtà: **Configurare una rete locale utilizzando router e switch reali o virtuali**

IPv4:

1. **Configurazione degli indirizzi IP e subnetting**: Gli studenti devono configurare gli indirizzi IP per ciascun dispositivo di rete nella rete locale utilizzando IPv4. Devono suddividere la rete in subnet e assegnare gli indirizzi IP in base a queste subnet, utilizzando il subnetting.
2. **Configurazione degli algoritmi di switching layer 2**: Gli studenti devono configurare gli switch reali o virtuali utilizzando algoritmi di switching layer 2 come STP (Spanning Tree Protocol) per gestire i loop di rete e assicurare la ridondanza.
3. **Configurazione dei protocolli di routing layer 3**: Gli studenti devono configurare i router reali o virtuali utilizzando protocolli di routing layer 3 come RIP (Routing Information Protocol) o OSPF (Open Shortest Path First) per instradare il traffico tra le subnet nella rete locale.

IPv6:

1. **Configurazione degli indirizzi IPv6**: Gli studenti devono configurare gli indirizzi IPv6 per ciascun dispositivo di rete nella rete locale utilizzando IPv6. Devono comprendere le differenze nella struttura degli indirizzi IPv6 rispetto a IPv4 e assegnare gli indirizzi in modo appropriato.
2. **Configurazione degli algoritmi di switching layer 2**: Gli studenti devono configurare gli switch reali o virtuali per supportare IPv6 e garantire il corretto funzionamento degli algoritmi di switching layer 2 anche per il nuovo protocollo.
3. **Configurazione dei protocolli di routing layer 3**: Gli studenti devono configurare i router reali o virtuali per supportare IPv6 e utilizzare protocolli di routing layer 3 come OSPFv3 per instradare il traffico IPv6 tra le subnet nella rete locale.

Gli studenti devono documentare tutte le configurazioni effettuate, compresi gli indirizzi IP, le subnet, le tabelle di routing e le configurazioni degli algoritmi di switching. Devono spiegare anche il funzionamento degli algoritmi di switching layer 2 e dei protocolli di routing layer 3 sia per IPv4 che per IPv6, evidenziando le differenze tra i due protocolli e le relative configurazioni.

Reti locali e reti geografiche: architettura fisica, sistemi operativi e programmi di comunicazione

Compito di realtà: Progettare una rete per la scuola. Gli studenti devono creare una piantina dettagliata che mostri la disposizione fisica dei dispositivi di rete (switch, router, access point, ecc.) e i cavi di collegamento. Devono anche scegliere e configurare il sistema operativo di rete (ad esempio, Windows Server, Linux) e i programmi di comunicazione necessari. Devono presentare il progetto e giustificare le loro scelte in base ai requisiti di rete.

Strumenti di simulazione di progettazione di reti locali

Compito di realtà: Utilizzare un simulatore di reti come Cisco Packet Tracer per progettare, configurare e testare una rete locale. Gli studenti devono creare un progetto che includa diverse subnet, VLAN, routing statico e dinamico, e configurazioni di sicurezza di base (ad esempio, ACL). Devono presentare il progetto e mostrare i risultati dei test di connettività e prestazioni della rete simulata.

Normative per il cablaggio strutturato standard EIA

Compito di realtà: Redigere un documento che descriva le normative per il cablaggio strutturato secondo gli standard EIA/TIA. Gli studenti devono includere diagrammi e descrizioni dettagliate delle categorie di cavi, connettori, schemi di cablaggio e pratiche di installazione. Devono poi creare un esempio pratico di cablaggio strutturato per una stanza o un edificio, seguendo le normative descritte.

Questi compiti di realtà permettono agli studenti di applicare le loro conoscenze teoriche a situazioni pratiche, migliorando così la loro comprensione delle reti di elaboratori e delle reti di comunicazione.

Metodologie e tecnologie per la sicurezza informatica

Compito di realtà: Creare una guida pratica per la sicurezza informatica per i compagni di classe o per la scuola. Gli studenti devono ricercare e sintetizzare informazioni su metodologie e tecnologie per la sicurezza informatica, come firewall, antivirus, IDS/IPS, crittografia, backup e ripristino dei dati, ecc. Devono quindi presentare questa guida in formato digitale o stampato, includendo suggerimenti pratici e best practices per proteggere i dati e i dispositivi.

Vulnerabilità, minacce e contromisure

Compito di realtà: Condurre un'analisi delle vulnerabilità di un sistema informatico. Gli studenti devono identificare le potenziali vulnerabilità in un sistema (ad esempio, un sito web, un'applicazione, un sistema operativo) utilizzando strumenti di scansione automatizzati e test manuali. Devono poi documentare le vulnerabilità trovate, valutare il rischio associato a ciascuna e suggerire contromisure per mitigare o eliminare i rischi.

Tecniche crittografiche e loro applicazioni

Compito di realtà: Implementare un sistema di comunicazione sicuro utilizzando tecniche di crittografia. Gli studenti devono creare un'applicazione che consenta a due utenti di scambiare messaggi crittografati utilizzando algoritmi di crittografia simmetrica (come AES) e asimmetrica (come RSA). Devono documentare il processo di sviluppo, spiegare i principi della crittografia utilizzata e dimostrare il funzionamento del sistema.

Controllo degli accessi

Compito di realtà: Progettare e implementare un sistema di controllo degli accessi per un'organizzazione scolastica o aziendale. Gli studenti devono definire ruoli e privilegi per diversi utenti (studenti, insegnanti, amministratori, ospiti) e implementare un sistema di autenticazione e autorizzazione che limiti l'accesso alle risorse in base a questi ruoli. Devono documentare il processo di progettazione e implementazione e testare il sistema con casi di prova.

Principali aspetti normativi

Compito di realtà: Condurre una ricerca sulle principali normative e leggi riguardanti la sicurezza informatica (come GDPR, HIPAA, ISO 27001). Gli studenti devono analizzare e spiegare i requisiti di conformità e le implicazioni per le organizzazioni che gestiscono dati sensibili. Devono inoltre discutere casi di violazione della sicurezza e le conseguenze legali per le aziende coinvolte.

Questi compiti di realtà offrono agli studenti l'opportunità di esplorare in profondità la sicurezza informatica, applicando le loro conoscenze teoriche a situazioni pratiche e attuali. Inoltre, li preparano ad affrontare sfide reali nel campo della sicurezza dei sistemi informatici e delle reti.

Progettazione concettuale, logica e fisica di una base di dati

Compito di realtà: Creare un progetto completo di una base di dati per un'applicazione di gestione di una biblioteca. Gli studenti devono iniziare con la progettazione

concettuale, identificando le entità, le relazioni e gli attributi necessari. Successivamente, devono tradurre il modello concettuale in uno schema logico e quindi in uno schema fisico, includendo l'ottimizzazione delle tabelle, gli indici e le vincolazioni di integrità referenziale.

Linguaggio SQL per l'interrogazione e la gestione di basi di dati

Compito di realtà: Scrivere e testare una serie di query SQL per estrarre informazioni da un database di esempio. Gli studenti devono creare un database relazionale che contenga dati realistici (ad esempio, dati di studenti, libri, prestiti) e quindi scrivere una serie di query SQL complesse per eseguire operazioni di interrogazione, aggiornamento, inserimento e cancellazione di dati. Devono testare le query e documentare i risultati ottenuti.

Strumenti per la progettazione e test di architetture 3-tier (WAMP, LAMP, XAMPP, EasyPHP)

Compito di realtà: Configurare e testare un'architettura web 3-tier utilizzando uno dei pacchetti di sviluppo (WAMP, LAMP, XAMPP, EasyPHP). Gli studenti devono installare il software sul proprio computer locale e creare una semplice applicazione web che utilizzi una base di dati MySQL (o MariaDB) come backend. Devono quindi testare l'applicazione, verificare la connessione al database e la corretta esecuzione delle operazioni CRUD (Create, Read, Update, Delete).

Questi compiti di realtà permettono agli studenti di applicare i concetti teorici della progettazione dei database e dell'interrogazione SQL in scenari pratici e attuali. Inoltre, offrono loro l'opportunità di acquisire esperienza pratica con gli strumenti di sviluppo web e di comprendere meglio l'architettura 3-tier nelle applicazioni web moderne.

Progettazione e sviluppo di applicazioni a tre livelli

Compito di realtà: Creare un'applicazione web a tre livelli per la gestione di un negozio online. Gli studenti devono progettare e sviluppare un frontend interattivo utilizzando HTML, CSS e JavaScript per la presentazione dei prodotti e l'interazione con gli utenti. Poi devono implementare un backend con un linguaggio di programmazione server-side (come PHP o Python) e un database relazionale (come MySQL) per la gestione dei dati dei prodotti, degli utenti e degli ordini. Infine, devono implementare un layer di accesso ai dati per la comunicazione tra il frontend e il backend. Devono documentare il processo di progettazione e sviluppo e testare l'applicazione per garantire la sua funzionalità e usabilità.

Sistemi multimediali

Compito di realtà: Creare una presentazione multimediale su un argomento scelto dagli studenti. Gli studenti devono utilizzare una varietà di formati multimediali, come testo, immagini, audio e video, per creare una presentazione coinvolgente e informativa.

Devono utilizzare strumenti di editing multimediale come Adobe Creative Suite o software open source come GIMP e Audacity per creare e modificare i contenuti multimediali. Devono quindi presentare la presentazione al resto della classe,

dimostrando la loro comprensione dei principi di progettazione multimediale e della capacità di utilizzare gli strumenti di creazione multimediale.

Rappresentazione digitale dei diversi tipi di informazione

Compito di realtà: Creare un progetto di digitalizzazione di una collezione di oggetti fisici, come libri, disegni o fotografie. Gli studenti devono utilizzare strumenti di digitalizzazione come scanner, fotocamere digitali o software di modellazione 3D per convertire gli oggetti fisici in formati digitali. Devono quindi organizzare e catalogare i dati digitalizzati utilizzando un sistema di gestione dei contenuti o un database. Devono presentare il progetto, spiegando il processo di digitalizzazione e le sfide affrontate durante il processo.

Componenti fisici per i sistemi multimediali

Compito di realtà: Costruire un dispositivo multimediale personalizzato utilizzando componenti hardware disponibili. Gli studenti devono selezionare e acquistare componenti come schede Arduino, Raspberry Pi, sensori, display e altoparlanti, quindi assemblare e programmare il dispositivo per eseguire funzionalità multimediali specifiche. Devono documentare il processo di progettazione, assemblaggio e programmazione e presentare il dispositivo finito alla classe, dimostrando le sue funzionalità e caratteristiche.

Strumenti di programmazione per i sistemi multimediali

Compito di realtà: Sviluppare un'applicazione ipertestuale utilizzando un linguaggio di programmazione specializzato orientato alle immagini. Gli studenti devono utilizzare uno strumento di sviluppo ipertestuale come Adobe Flash o Adobe Animate per creare un'applicazione interattiva che integri immagini, testo e animazioni. Devono utilizzare azioni di scripting per aggiungere interattività alla loro applicazione, come la navigazione tra pagine, la visualizzazione di contenuti multimediali e l'interazione con gli utenti. Devono testare e ottimizzare l'applicazione per garantire la sua funzionalità e compatibilità con diversi dispositivi e browser.

Strumenti di editoria digitale multimediale

Compito di realtà: Progettare e creare una rivista digitale multimediale su un argomento di interesse degli studenti. Gli studenti devono utilizzare strumenti di editoria digitale come Adobe InDesign, Microsoft Publisher o software open source come Scribus per progettare il layout della rivista e aggiungere contenuti multimediali come immagini, audio, video e link ipertestuali. Devono quindi esportare la rivista in formato digitale e condividerla con il resto della classe o pubblicarla online. Devono riflettere sul processo di progettazione e creazione e sulla loro esperienza nell'utilizzo degli strumenti di editoria digitale multimediale.

Questi compiti di realtà offrono agli studenti l'opportunità di sviluppare competenze pratiche nella progettazione e nello sviluppo di applicazioni a tre livelli e nei sistemi multimediali, utilizzando una varietà di strumenti e tecnologie. Inoltre, incoraggiano la creatività e l'innovazione nell'utilizzo dei media digitali per comunicare e condividere informazioni.

Studi di fattibilità di progetti informatici

Compito di realtà: Svolgere uno studio di fattibilità per un progetto software. Gli studenti devono identificare e analizzare i requisiti del progetto, valutarne la fattibilità tecnica ed economica e valutare i rischi associati. Devono redigere un business plan che includa una descrizione del progetto, l'analisi di mercato, la pianificazione finanziaria e il calcolo del ROI (Return on Investment).

Definizione di un budget di un progetto software

Compito di realtà: Definire un budget per un progetto software. Gli studenti devono identificare e stimare i costi associati al progetto, inclusi costi di personale, hardware, software, formazione e altri costi operativi. Devono anche valutare i ricavi previsti dal progetto e calcolare il ROI. Devono presentare il budget in forma tabellare e grafica, illustrando in dettaglio le voci di spesa e le fonti di finanziamento.

Costruzione di WBS, PBS e PERT

Compito di realtà: Costruire una struttura di scomposizione del lavoro (WBS), una struttura di scomposizione del prodotto (PBS) e un diagramma PERT per un progetto software. Gli studenti devono identificare le attività del progetto, suddividerle in sottoattività e organizzarle in una gerarchia logica utilizzando la WBS e la PBS. Devono quindi utilizzare il diagramma PERT per rappresentare la sequenza delle attività, le dipendenze e le stime di durata. Devono anche calcolare il percorso critico del progetto.

Diagramma di GANTT

Compito di realtà: Creare un diagramma di Gantt per la pianificazione e il monitoraggio di un progetto software. Gli studenti devono elencare le attività del progetto, assegnare loro una durata e stabilire le dipendenze tra di esse. Devono quindi rappresentare le attività e le loro durate su un grafico a barre temporali utilizzando un software di gestione di progetto o un'applicazione online. Devono aggiungere le milestone e le scadenze del progetto e utilizzare il diagramma di Gantt per monitorare lo stato di avanzamento del progetto.

Questi compiti di realtà offrono agli studenti l'opportunità di acquisire competenze pratiche nella gestione d'impresa e nella gestione dei progetti informatici, consentendo loro di applicare concetti teorici a scenari reali e di sviluppare capacità di pianificazione, analisi e controllo dei progetti.

Firma digitale, PEC, identità digitale, SPID, enti certificatori, marche temporali

Compito di realtà: **Creazione di una guida informativa sulla sicurezza digitale**

1. **Ricerca e studio**: Gli studenti devono condurre una ricerca approfondita sui concetti di firma digitale, PEC, identità digitale, SPID, enti certificatori e marche temporali. Devono comprendere come funzionano, perché sono importanti e quali sono le implicazioni per la sicurezza digitale.

2. **Creazione della guida**: Gli studenti devono creare una guida informativa sulla sicurezza digitale che illustri in modo chiaro e accessibile i concetti sopra menzionati. La guida dovrebbe includere spiegazioni dettagliate di ciascun concetto, esempi pratici, consigli per l'uso sicuro e riferimenti a risorse aggiuntive.
3. **Elaborazione visiva**: Gli studenti possono arricchire la guida con elementi visivi, come infografiche, diagrammi e grafici, per rendere i concetti più comprensibili e coinvolgenti per il pubblico.
4. **Divulgazione della guida**: Gli studenti devono diffondere la guida tra i loro compagni di classe, insegnanti, genitori e altri membri della comunità scolastica. Possono organizzare una presentazione o un evento di divulgazione per condividere le informazioni contenute nella guida e sensibilizzare sulla sicurezza digitale.
5. **Feedback e valutazione**: Gli studenti possono raccogliere feedback sulla guida dai destinatari e valutare l'efficacia della comunicazione dei concetti di sicurezza digitale. Possono utilizzare questo feedback per apportare eventuali miglioramenti alla guida e riflettere sull'esperienza di divulgazione.

Questo compito offre agli studenti l'opportunità di approfondire la comprensione dei concetti di sicurezza digitale e allo stesso tempo sviluppare competenze di ricerca, comunicazione e divulgazione. Inoltre, promuove la consapevolezza sulla sicurezza digitale nella comunità scolastica e oltre.

Protezione dati personali, Privacy e sicurezza

Compito di realtà: Condurre una campagna di sensibilizzazione sulla protezione dei dati personali e la privacy online. Gli studenti devono creare materiali educativi, come brochure, poster e video, che illustrino le migliori pratiche per proteggere i dati personali online, prevenire il furto di identità e navigare in modo sicuro su Internet. Devono diffondere questi materiali all'interno della scuola e della comunità locale e organizzare eventi informativi su questi temi.

E-Governance e Amministrazione Digitale

Compito di realtà: Analizzare e valutare il livello di digitalizzazione dei servizi pubblici locali. Gli studenti devono condurre ricerche e interviste per valutare l'accessibilità e l'efficacia dei servizi online offerti dalla pubblica amministrazione locale. Devono identificare le aree di miglioramento e proporre soluzioni per promuovere l'e-governance e migliorare l'amministrazione digitale, ad esempio attraverso la semplificazione dei processi, l'implementazione di nuovi servizi online o l'ottimizzazione delle piattaforme esistenti.

Gestione documentale e conservazione dei documenti digitali

Compito di realtà: Creare un sistema di gestione documentale per la scuola o un'associazione locale. Gli studenti devono selezionare e configurare un software di gestione documentale open source o commerciale e implementare un sistema che consenta di organizzare, archiviare e recuperare documenti digitali in modo efficiente e sicuro. Devono fornire formazione al personale sull'utilizzo del sistema e garantire la conformità alle normative sulla conservazione dei documenti.

Normative e regolamenti sulla trasformazione digitale

Compito di realtà: Condurre una ricerca sulle principali normative e regolamenti relativi alla trasformazione digitale in Italia e nell'Unione Europea. Gli studenti devono esaminare leggi come il GDPR (General Data Protection Regulation), la direttiva eIDAS (Electronic Identification and Trust Services), e le normative nazionali sull'amministrazione digitale. Devono identificare le principali disposizioni normative, le implicazioni per cittadini, imprese e pubblica amministrazione e le sfide nell'attuazione.

Questi compiti di realtà offrono agli studenti l'opportunità di comprendere in profondità la trasformazione digitale e i suoi impatti sulla società, nonché di sviluppare competenze pratiche nella gestione dei dati digitali, nella sicurezza informatica e nella conformità normativa. Inoltre, promuovono la consapevolezza e la responsabilità nell'utilizzo delle tecnologie digitali.