MODUL AJAR DEEP LEARNING
MATA PELAJARAN : INFORMATIKA
BAB 2: STRATEGI ALGORITMA DAN PEMROGRAMAN

A. IDENTITAS MODUL

Nama Sekolah	**:**	.....
Nama Penyusun	**:**	.....
Mata Pelajaran	**:**	**Informatika**
Kelas / Fase /Semester	**:**	**XI/ F / Ganjil**
Alokasi Waktu	**:**	
Tahun Pelajaran	**:**	**20.. / 20..**

B. IDENTIFIKASI KESIAPAN PESERTA DIDIK

Peserta didik kelas XI diasumsikan telah memiliki pemahaman dasar tentang konsep algoritma dari jenjang sebelumnya, seperti kemampuan untuk mengidentifikasi masalah, menyusun langkah-langkah penyelesaian sederhana, dan mungkin sedikit pengenalan terhadap konsep pemrograman visual sederhana (misalnya dengan Scratch atau Blockly). Keterampilan yang sudah dimiliki meliputi berpikir logis, kemampuan memecahkan masalah dasar, dan mungkin telah mencoba membuat program sederhana. Namun, pemahaman tentang efisiensi algoritma, debugging yang kompleks, dan perancangan program yang terstruktur mungkin masih terbatas dan perlu dikembangkan lebih lanjut. Minat terhadap dunia digital dan teknologi umumnya tinggi, namun motivasi untuk belajar pemrograman yang lebih mendalam bisa bervariasi.

C. KARAKTERISTIK MATERI PELAJARAN

Materi "Strategi Algoritma dan Pemrograman" merupakan jenis pengetahuan konseptual, prosedural, dan metakognitif. Peserta didik akan memahami konsep dasar algoritma dan pemrograman, mampu menerapkan berbagai strategi penyelesaian masalah melalui algoritma, dan merefleksikan proses berpikir mereka dalam merancang solusi. Materi ini sangat relevan dengan kehidupan nyata peserta didik, karena keterampilan berpikir algoritmik dan pemrograman adalah dasar dari banyak inovasi teknologi yang mereka gunakan sehari-hari, serta relevan untuk pengembangan logika dan pemecahan masalah dalam berbagai aspek kehidupan. Tingkat kesulitan materi ini bersifat moderat hingga tinggi, membutuhkan kemampuan berpikir abstrak dan ketelitian. Struktur materi akan disajikan secara bertahap, mulai dari pengenalan strategi, implementasi dalam pemrograman, hingga debugging dan optimasi. Integrasi nilai dan karakter akan dilakukan melalui penekanan pada ketelitian, kesabaran, kerja sama dalam memecahkan masalah, serta sikap pantang menyerah dalam menghadapi kesalahan (bug).

D DIMENSI PROFIL LULUSAN PEMBELAJARAN

Berdasarkan tujuan pembelajaran Bab 2: Strategi Algoritma dan Pemrograman, dimensi profil lulusan yang akan dicapai adalah:

- **Penalaran Kritis:** Peserta didik mampu menganalisis masalah, merancang algoritma yang efisien, dan mengidentifikasi serta memperbaiki kesalahan (bug) dalam program.

- **Kreativitas:** Peserta didik mampu mengembangkan berbagai strategi algoritmik untuk menyelesaikan masalah yang berbeda dan menciptakan solusi program yang inovatif.
- **Kolaborasi:** Peserta didik aktif bekerja sama dalam kelompok untuk merancang, menguji, dan memperbaiki program.
- **Kemandirian:** Peserta didik mampu secara mandiri merancang algoritma, menulis kode, dan melakukan debugging.
- **Komunikasi:** Peserta didik mampu menjelaskan strategi algoritmik dan cara kerja program mereka secara lisan dan tertulis.

DESAIN PEMBELAJARAN

A. CAPAIAN PEMBELAJARAN (CP) NOMOR : 32 TAHUN 2024

Di akhir fase ini (Fase F, kelas XI), peserta didik diharapkan mampu:

- Menganalisis masalah dan mengidentifikasi kebutuhan komputasi untuk menyelesaikannya.
- Menerapkan strategi algoritmik yang efektif untuk menyelesaikan masalah komputasi.
- Menulis program sederhana menggunakan bahasa pemrograman visual atau berbasis teks untuk mengimplementasikan algoritma.
- Melakukan debugging dan menguji program untuk memastikan kebenaran dan efisiensinya.
- Berpikir secara komputasional untuk memecahkan masalah dari berbagai disiplin ilmu.

B. LINTAS DISIPLIN ILMU YANG RELEVAN

- **Matematika:** Konsep logika, deret, pola, dan struktur data dasar sangat terkait dengan algoritma.
- **Fisika/Kimia/Biologi:** Pemecahan masalah dalam eksperimen atau simulasi seringkali membutuhkan pendekatan algoritmik.
- **Desain/Seni:** Perancangan antarmuka pengguna (UI) dan pengalaman pengguna (UX) dalam aplikasi pemrograman.
- **Ilmu Sosial/Ekonomi:** Penerapan algoritma dalam analisis data atau simulasi model ekonomi/sosial.

C. TUJUAN PEMBELAJARAN

Pertemuan 1: Pengenalan Strategi Algoritma & Konsep Dasar Pemrograman

- Peserta didik dapat mengidentifikasi masalah yang dapat diselesaikan dengan algoritma setelah mengamati studi kasus sederhana.
- Peserta didik dapat menjelaskan konsep dasar algoritma dan flowchart dengan tepat setelah diskusi dan eksplorasi contoh.
- Peserta didik dapat menerapkan salah satu strategi algoritma dasar (misalnya, divide and conquer atau greedy algorithm) untuk menyelesaikan masalah sederhana yang diberikan secara mandiri.

Pertemuan 2: Implementasi Algoritma dengan Bahasa Pemrograman Visual/Teks Sederhana

- Peserta didik dapat menulis kode program sederhana (misalnya, menggunakan Python atau Blockly) untuk mengimplementasikan algoritma yang telah dirancang dengan bantuan guru.
- Peserta didik dapat mengidentifikasi sintaks dasar dan struktur program (misalnya, variabel, *input/output*, *loop*, *conditional statement*) dalam contoh program.
- Peserta didik dapat menguji program sederhana dan mengidentifikasi kesalahan dasar (*syntax error* atau *logic error*) secara mandiri.

Pertemuan 3 & 4: Pemrograman Berbasis Proyek: Merancang dan Membuat Aplikasi Sederhana

- Peserta didik dapat merancang algoritma yang lebih kompleks untuk proyek aplikasi sederhana (misalnya, kalkulator sederhana, game tebak angka, atau aplikasi daftar belanja).
- Peserta didik dapat menulis dan mengimplementasikan kode program untuk proyek mereka secara kolaboratif dalam kelompok.
- Peserta didik dapat melakukan debugging secara sistematis untuk memperbaiki kesalahan dalam program yang mereka buat.

Pertemuan 5: Optimasi dan Presentasi Proyek

- Peserta didik dapat mengidentifikasi potensi optimasi dalam algoritma atau kode program mereka.
- Peserta didik dapat mempresentasikan proyek aplikasi sederhana mereka di depan kelas, menjelaskan algoritma dan cara kerja program.
- Peserta didik dapat memberikan umpan balik konstruktif terhadap proyek teman sebaya.

D. TOPIK PEMBELAJARAN KONTEKSTUAL

Topik pembelajaran akan berpusat pada "Problem Solving with Algorithms: From Everyday Life to Digital Solutions." Peserta didik akan diajak untuk melihat bagaimana algoritma tidak hanya ada dalam dunia komputer, tetapi juga dalam rutinitas sehari-hari mereka. Mereka akan mengidentifikasi masalah-masalah kecil di sekitar mereka dan mencoba merancang solusi algoritmik untuknya, yang kemudian dapat diimplementasikan dalam bentuk program sederhana. Contoh topik kontekstual: "Merancang Algoritma untuk Mencari Barang Hilang," "Membuat Program Pengatur Jadwal Belajar," atau "Membangun Simulasi Permainan Sederhana."

E. KERANGKA PEMBELAJARAN

PRAKTIK PEDAGOGIK:

- **Metode Pembelajaran Berbasis Proyek:** Peserta didik akan terlibat dalam proyek merancang dan mengimplementasikan program sederhana. Proyek ini akan melibatkan tahapan analisis masalah, perancangan algoritma, penulisan kode, debugging, dan presentasi.
- **Diskusi Kelompok:** Digunakan untuk menganalisis masalah, mendiskusikan berbagai strategi algoritma, memecahkan masalah pemrograman bersama, dan memberikan umpan balik antar-teman.
- **Eksplorasi Lapangan (Virtual):** Jika memungkinkan, mengunjungi secara virtual situs-situs yang menyediakan studi kasus masalah algoritmik nyata (misalnya, bagaimana Google Maps bekerja atau algoritma rekomendasi di platform streaming).
- **Wawancara (Antar-kelompok):** Peserta didik dapat melakukan wawancara singkat antar-kelompok untuk mendapatkan umpan balik terhadap algoritma atau draf kode mereka.
- **Presentasi:** Peserta didik akan mempresentasikan proyek program mereka di akhir unit.

MITRA PEMBELAJARAN:

- **Lingkungan Sekolah:** Laboratorium komputer (jika tersedia), guru mata pelajaran

lain (untuk integrasi lintas disiplin, misalnya guru matematika untuk logika).

- **Lingkungan Luar Sekolah:** Komunitas *developer* lokal (jika ada kesempatan untuk mengundang narasumber atau *mentor* virtual), platform pembelajaran daring (Coursera, Codecademy, freeCodeCamp).
- **Masyarakat:** Mengidentifikasi masalah-masalah sederhana di lingkungan sekitar yang dapat diselesaikan dengan pendekatan komputasi.

LINGKUNGAN BELAJAR:

- **Ruang Fisik:** Kelas diatur untuk memfasilitasi kerja kelompok dan akses ke komputer. Tersedia papan tulis/layar, dan proyektor.
- **Ruang Virtual:** Pemanfaatan *online IDE* (Integrated Development Environment), platform pembelajaran daring (Google Classroom, Replit, PythonAnywhere, atau platform lain yang sesuai), dan forum diskusi.
  - **Budaya Belajar:**
- **Kolaboratif:** Mendorong kerja sama tim, saling membantu dalam debugging, dan berbagi solusi.
- **Berpartisipasi Aktif:** Mendorong setiap peserta didik untuk mencoba, bereksperimen, dan tidak takut membuat kesalahan.
- **Rasa Ingin Tahu:** Membangkitkan minat peserta didik untuk menjelajahi berbagai strategi algoritma dan bahasa pemrograman.

PEMANFAATAN DIGITAL:

- **Perpustakaan Digital:** Menggunakan sumber daya daring seperti dokumentasi bahasa pemrograman, tutorial coding, dan contoh-contoh algoritma.
- **Forum Diskusi Daring:** Menggunakan fitur forum di Google Classroom atau platform lain untuk diskusi asinkron, berbagi potongan kode, dan bertanya tentang masalah debugging.
- **Penilaian Daring:** Menggunakan Google Forms atau platform kuis daring (Quizizz, Kahoot) untuk asesmen formatif.
- **Kahoot/Mentimeter:** Digunakan untuk aktivitas pemanasan, kuis singkat tentang konsep algoritma, atau survei cepat tentang pemahaman.
- **Google Classroom:** Sebagai pusat pengelolaan kelas, berbagi materi (termasuk *link* ke *online IDE*), pengumpulan tugas, dan pengumuman.
- **Online IDE/Code Editor:** Platform seperti Replit, CodePen, atau bahkan Google Colab untuk menulis, menjalankan, dan menguji kode program secara daring.

F. LANGKAH-LANGKAH PEMBELAJARAN BERDIFERENSIASI

PERTEMUAN 1: PENGENALAN STRATEGI ALGORITMA & KONSEP DASAR PEMROGRAMAN

KEGIATAN PENDAHULUAN (MINDFUL LEARNING, JOYFUL LEARNING):

- **Pemanasan (Joyful):** Guru menampilkan video singkat tentang bagaimana algoritma digunakan dalam kehidupan sehari-hari (misalnya, algoritma mencari jalur di Google Maps, algoritma sortir daftar lagu). Pertanyaan pemicu: "Apakah kalian pernah memikirkan 'langkah-langkah' yang dilakukan aplikasi ini?"
- **Aktivasi Pengetahuan Awal (Mindful):** Guru bertanya: "Apa yang kalian pahami tentang algoritma? Berikan contoh langkah-langkah dalam kehidupan sehari-hari yang merupakan sebuah algoritma." (Diferensiasi Konten: Memberikan contoh yang

lebih sederhana atau lebih kompleks sesuai level pemahaman awal).

- **Membangun Koneksi (Meaningful):** Guru menghubungkan algoritma dengan pemecahan masalah. "Mengapa penting bagi kita untuk memahami bagaimana cara 'memerintahkan' komputer untuk melakukan sesuatu?"
- **Orientasi (Mindful):** Menyampaikan tujuan pembelajaran hari ini dan kriteria keberhasilan.

KEGIATAN INTI (MEANINGFUL LEARNING, JOYFUL LEARNING):

Eksplorasi Konsep (Memahami):

- Guru memaparkan konsep algoritma, flowchart, dan notasi pseudocode dengan contoh-contoh sederhana (misalnya, algoritma memasak mi instan, algoritma mencari buku di perpustakaan).
- **Berkesadaran (Mindful):** Peserta didik diminta untuk memvisualisasikan setiap langkah dalam algoritma yang dijelaskan.
- **Studi Kasus Sederhana (Meaningful):** Guru memberikan masalah nyata sederhana (misalnya, bagaimana menyortir tumpukan baju, bagaimana menemukan angka terbesar dalam daftar).

Aplikasi Strategi (Mengaplikasi):

- **Diskusi Kelompok (Kolaborasi):** Dalam kelompok, peserta didik mendiskusikan berbagai cara untuk menyelesaikan masalah studi kasus tersebut, dan kemudian memilih salah satu strategi (misalnya, *greedy algorithm* atau *divide and conquer* sederhana).
- **Merancang Flowchart/Pseudocode (Meaningful):** Peserta didik secara kolaboratif membuat flowchart atau menulis pseudocode untuk algoritma yang mereka rancang. Guru menyediakan *template* atau contoh untuk membantu. (Diferensiasi Proses: Kelompok yang lebih mahir dapat diminta untuk mempertimbangkan efisiensi, sementara kelompok yang membutuhkan bantuan dapat fokus pada kelengkapan langkah).

Refleksi (Merefleksi, Berkesadaran):

- Setiap kelompok mempresentasikan flowchart/pseudocode mereka dan menjelaskan mengapa mereka memilih strategi tersebut.
- Guru memberikan umpan balik konstruktif dan menyoroti berbagai pendekatan yang mungkin.
- **Refleksi Singkat (Individu):** Peserta didik menuliskan satu tantangan dalam merancang algoritma dan satu hal yang paling menarik.

PERTEMUAN 2: IMPLEMENTASI ALGORITMA DENGAN BAHASA PEMROGRAMAN VISUAL/TEKS SEDERHANA

- **Kegiatan Inti (Meaningful Learning, Joyful Learning):**

Eksplorasi Sintaks (Memahami):

- Guru memperkenalkan *online IDE* dan sintaks dasar bahasa pemrograman yang akan digunakan (misalnya, Python: *print(), input(), if/else, for loop*).
- **Demonstrasi Langsung (Joyful):** Guru mendemonstrasikan penulisan dan eksekusi program sederhana secara langsung.
- **Latihan Terpandu (Meaningful):** Peserta didik mengikuti demonstrasi guru untuk

menulis dan menjalankan program-program sederhana yang diberikan.

Implementasi Algoritma (Mengaplikasi):

- **Coding Challenge (Joyful):** Guru memberikan serangkaian "tantangan coding" kecil yang mengimplementasikan algoritma sederhana dari Pertemuan 1 (misalnya, program yang menghitung luas persegi panjang, program yang menentukan apakah sebuah angka genap atau ganjil).
- **Debugging Dasar (Berkesadaran):** Peserta didik diminta untuk secara sengaja membuat kesalahan (*syntax error*) dan kemudian memperbaikinya, atau menganalisis *logic error* sederhana. Guru membimbing mereka dalam proses debugging. (Diferensiasi Proses: Menyediakan *error message* yang lebih jelas atau petunjuk *debugging* terstruktur untuk kelompok yang kesulitan).

Refleksi (Merefleksi, Berkesadaran):

- Peserta didik berbagi program yang telah mereka buat dan tantangan yang mereka hadapi selama proses coding dan debugging.
- **Pair Programming (Kolaborasi):** Peserta didik secara berpasangan saling mengulas kode satu sama lain dan memberikan saran perbaikan.

PERTEMUAN 3 & 4: PEMROGRAMAN BERBASIS PROYEK: MERANCANG DAN MEMBUAT APLIKASI SEDERHANA

KEGIATAN INTI (MEANINGFUL LEARNING, JOYFUL LEARNING):

Identifikasi Masalah Proyek (Kreativitas, Meaningful):

- Guru memandu diskusi untuk mengidentifikasi masalah-masalah yang lebih kompleks yang dapat diselesaikan dengan program sederhana (misalnya, membuat "aplikasi" penghitung BMI, "game" tebak angka, "aplikasi" daftar belanja).
- **Brainstorming Ide (Joyful):** Peserta didik secara berkelompok *brainstorming* ide proyek mereka. (Diferensiasi Produk: Memberikan daftar ide proyek yang sudah ditentukan untuk kelompok yang membutuhkan panduan lebih, atau membiarkan kelompok berkreasi penuh).

Perencanaan Proyek (Mengaplikasi, Mandiri):

- Dalam kelompok, peserta didik merancang algoritma untuk proyek mereka, termasuk membuat flowchart yang lebih rinci atau pseudocode yang terstruktur.
- Guru menyediakan *template* perencanaan proyek dan rubrik penilaian proyek.
- **Wawancara Antar-kelompok:** Kelompok saling mempresentasikan rencana proyek mereka dan mendapatkan umpan balik awal.

Pengembangan Kode (Meaningful, Berkesadaran):

- Peserta didik mulai menulis kode program untuk proyek mereka di *online IDE*.
- Guru berkeliling, memberikan bimbingan individual dan umpan balik formatif, terutama dalam debugging.
- **Strategi Debugging (Berkesadaran):** Mendorong peserta didik untuk mencatat kesalahan yang terjadi dan strategi yang digunakan untuk memperbaikinya.
- **Peer Debugging (Kolaborasi):** Mendorong anggota kelompok untuk saling membantu dalam menemukan dan memperbaiki bug.

PERTEMUAN 5: OPTIMASI DAN PRESENTASI PROYEK

KEGIATAN INTI (MEANINGFUL LEARNING, JOYFUL LEARNING):

Optimasi Kode (Meaningful, Berkesadaran):

- Guru memperkenalkan konsep efisiensi sederhana (misalnya, menghindari pengulangan kode, penggunaan fungsi).
- Peserta didik diminta untuk meninjau kembali kode program mereka dan mengidentifikasi bagian yang dapat dioptimasi. (Diferensiasi Proses: Guru dapat memberikan contoh optimasi untuk kelompok yang kesulitan, atau menantang kelompok yang lebih mahir untuk mencari cara optimasi yang lebih kompleks).

Produksi Akhir dan Persiapan Presentasi (Kreativitas, Joyful):

- Peserta didik menyelesaikan program mereka dan mempersiapkan presentasi (misalnya, membuat slide yang menjelaskan algoritma, demo program).
  - **Presentasi Proyek (Komunikasi, Kolaborasi):**
- Setiap kelompok mempresentasikan aplikasi sederhana mereka di depan kelas.
- Mereka menjelaskan masalah yang ingin diselesaikan, algoritma yang digunakan, dan mendemonstrasikan cara kerja program.
- **Sesi Tanya Jawab dan Umpan Balik (Meaningful):** Peserta didik lain dapat mengajukan pertanyaan dan memberikan umpan balik konstruktif.

KEGIATAN PENUTUP (MEMBERIKAN UMPAN BALIK, MENYIMPULKAN, PERENCANAAN SELANJUTNYA):

- **Umpan Balik Konstruktif (Berkesadaran):** Guru memberikan umpan balik secara keseluruhan tentang pembelajaran unit ini, mengapresiasi upaya peserta didik dalam merancang algoritma dan menulis program, serta menyoroti area peningkatan.
- **Menyimpulkan Pembelajaran (Meaningful):** Guru bersama peserta didik menyimpulkan poin-poin penting yang telah dipelajari tentang strategi algoritma, konsep pemrograman, dan proses debugging. "Apa pelajaran terbesar yang kalian dapatkan tentang cara berpikir seperti seorang programmer?"
- **Perencanaan Pembelajaran Selanjutnya (Mindful, Partisipatif):** Guru mengajak peserta didik untuk memberikan masukan tentang topik atau bahasa pemrograman yang ingin mereka pelajari di unit berikutnya, atau bagaimana mereka dapat melanjutkan belajar coding di luar kelas. "Bagaimana kalian bisa terus mengasah keterampilan pemrograman kalian setelah unit ini?"
- **Apresiasi (Joyful):** Guru memberikan apresiasi atas partisipasi dan kerja keras seluruh peserta didik.

G. ASESMEN PEMBELAJARAN

ASESMEN AWAL PEMBELAJARAN

- **Observasi:** Mengamati partisipasi peserta didik dalam diskusi awal tentang algoritma dalam kehidupan sehari-hari, tingkat kepercayaan diri mereka dalam memecahkan masalah.
- **Wawancara Singkat:** Guru dapat bertanya tentang pengalaman mereka dengan algoritma atau program sederhana yang pernah mereka lihat/gunakan.
- *Contoh Soal/Pertanyaan Wawancara:* "Apa yang kamu pahami tentang algoritma? Bisakah kamu memberikan contoh langkah-langkah yang kamu lakukan untuk menyelesaikan masalah sederhana di rumah?"
- **Kuesioner:** Memberikan kuesioner singkat tentang pemahaman dasar algoritma dan minat terhadap pemrograman.

SOAL KUESIONER:

1. Jelaskan dengan kata-kata sendiri apa itu algoritma.
2. Sebutkan dua contoh algoritma yang kamu temukan dalam kehidupan sehari-hari.
3. Apakah kamu tertarik untuk belajar bagaimana komputer menyelesaikan masalah? (Ya/Tidak/Mungkin)
4. Seberapa percaya diri kamu dalam memecahkan masalah yang memerlukan langkah-langkah logis? (Skala 1-5)
5. Apa yang kamu harapkan dari pembelajaran Informatika tentang pemrograman?

- **Tes Diagnostik:** Tes singkat berupa soal pilihan ganda atau esai singkat untuk mengukur pemahaman konsep dasar algoritma dan logika.

SOAL TES DIAGNOSTIK:

1. Manakah dari berikut ini yang paling menggambarkan sebuah algoritma? a. Sebuah program komputer yang sudah jadi b. Sekumpulan instruksi yang jelas dan terurut untuk menyelesaikan masalah c. Sebuah perangkat lunak yang diinstal di komputer d. Hasil akhir dari sebuah perhitungan
2. Susunlah langkah-langkah (algoritma) untuk membuat secangkir teh panas.
3. Apa perbedaan utama antara flowchart dan pseudocode?
4. Jika ada dua variabel, A = 10 dan B = 5. Tuliskan algoritma untuk menukar nilai A dan B tanpa menggunakan variabel ketiga.
5. Menurutmu, mengapa debugging itu penting dalam pemrograman?

ASESMEN PROSES PEMBELAJARAN

TUGAS HARIAN:

- **Perancangan Flowchart/Pseudocode:** Peserta didik menyerahkan flowchart atau pseudocode untuk masalah tertentu.
- *Contoh Soal Tugas Harian:* "Buatlah flowchart untuk algoritma yang menentukan apakah sebuah bilangan yang dimasukkan pengguna adalah bilangan prima atau bukan."
    - **Latihan Coding Sederhana:** Peserta didik mengirimkan *screenshot* atau *link* ke kode program sederhana yang mereka tulis di *online IDE*.
- *Contoh Soal Tugas Harian:* "Tulis program Python (atau menggunakan Blockly) yang meminta pengguna memasukkan nama mereka, lalu mencetak 'Halo, [Nama Pengguna]! Selamat datang di dunia pemrograman!'"

DISKUSI KELOMPOK:

- **Rubrik Observasi Diskusi:** Guru menggunakan rubrik untuk mengamati partisipasi, kualitas argumen, dan kemampuan kolaborasi peserta didik selama diskusi (misalnya, saat memecahkan masalah debugging bersama).
- *Contoh Soal/Prompt Diskusi:* "Diskusikan dengan kelompokmu: Apa strategi terbaik untuk menemukan dan memperbaiki kesalahan (bug) dalam sebuah program yang sudah sangat panjang?"
- **Presentasi Rencana Proyek:** Kelompok mempresentasikan rencana algoritma untuk proyek mereka.

- *Contoh Soal/Prompt Presentasi:* "Presentasikan ide proyek kelompokmu, jelaskan masalah apa yang ingin kalian selesaikan, dan bagaimana algoritma utama program kalian akan bekerja."

ASESMEN AKHIR PEMBELAJARAN

- **Jurnal Reflektif (Individu):** Peserta didik menulis jurnal reflektif tentang pengalaman mereka dalam merancang algoritma dan pemrograman.

SOAL JURNAL REFLEKTIF:

1. Apa tantangan terbesar yang kamu hadapi saat mencoba menulis kode program dan bagaimana kamu mengatasinya?
2. Apa yang kamu pelajari tentang pentingnya perencanaan (misalnya, membuat flowchart/pseudocode) sebelum menulis kode?
3. Bagaimana pemahamanmu tentang algoritma telah berubah setelah unit ini?
4. Sebutkan satu keterampilan baru yang kamu dapatkan dalam unit ini yang menurutmu akan berguna di masa depan.
5. Jika kamu bisa memberi saran kepada seseorang yang baru belajar pemrograman, apa yang akan kamu katakan?

TES TERTULIS (STUDI KASUS/ANALISIS KODE):

SOAL TES TERTULIS:

1. Diberikan sebuah masalah: "Hitung rata-rata nilai dari 5 mata pelajaran." Tuliskan algoritma untuk masalah ini dalam bentuk pseudocode.
2. Perhatikan kode program berikut:
   - Python angka = 10
   - if angka % 2 == 0:
   - print("Genap") else: print("Ganjil")
3. Jelaskan apa yang akan dicetak oleh program ini dan mengapa.
4. Jelaskan perbedaan antara *syntax error* dan *logic error* dalam pemrograman, dan berikan contoh masing-masing.
5. Mengapa efisiensi algoritma itu penting, terutama untuk masalah yang besar? Berikan contoh di mana efisiensi sangat krusial.
6. Bagaimana cara berpikir komputasional dapat membantumu memecahkan masalah di luar konteks pemrograman? Berikan satu contoh nyata.

TUGAS AKHIR/PROYEK:

- **Proyek Aplikasi Sederhana:** Penilaian terhadap program yang dihasilkan peserta didik, berdasarkan rubrik yang mencakup: fungsionalitas, kelengkapan algoritma, struktur kode, kemampuan *debugging*, dan inovasi.
- *Contoh Soal Tugas Akhir:* "Buatlah sebuah program sederhana (misalnya, kalkulator BMI, game tebak angka dengan batas percobaan, atau aplikasi konverter suhu) menggunakan bahasa pemrograman yang telah dipelajari. Sertakan flowchart/pseudocode algoritmanya."
- **Presentasi Proyek Akhir:** Penilaian terhadap kemampuan presentasi, kejelasan penjelasan algoritma, dan demonstrasi program.

- *Contoh Soal Presentasi Proyek:* "Presentasikan proyek program kelompokmu. Jelaskan algoritma yang kalian gunakan, tunjukkan demonstrasi program, dan jelaskan tantangan yang kalian hadapi serta solusinya."