LISTA DE SUBIECTE PENTRU EVALUAREA FINALĂ SDA

Tema 1.

Tipul de date pointer. Alocare dinamică a memoriei. Funcții predefinite.

Tema 2.

Sortarea datelor. Algoritmi de sortare şi analiza performanțelor. Tehnici și metode de sortare: 1) Bubble Sort; 2) Selection Sort; 3) Insertion Sort; 4) Merge Sort; 5) Quick Sort; 6) Shell Sort; 7) Counting Sort; 8) Comb Sort; 9) Radix Sort.

Tema 3.

Sortarea datelor. Heap Sort. Aplicarea dinamică arbori. Alocarea de memorie. Clasificarea arborilor dinamici. Arbori binari. Operații asupra arborilor binari.

Tema 4.

Structuri statice de date *struct*, *union*, cu câmpuri de *biţi*. Declararea, inițializarea, citirea şi afișarea structurilor, aplicarea pointerilor. Structuri dinamice de date. Alocare de memorie. Elaborarea programelor în C, care operează cu diferite structure.

Tema 5.

Tipul de date FILE. Fișiere de tip logic şi fizic. Funcții predefinite pentru date de tip FILE: *fopen(), fread(), fwrite(), fprintf(), fscanf(), fseek(), feof()*. Operații cu fișiere: creare, afișare, adăugare, corectare, sortare. Acces secvențial şi direct la fișiere.

Tema 6.

Aplicații dinamice: stiva, şir de așteptare, listă liniară, lista înlănțuită, listă bidirecțională. Operații asupra stivelor, șirurilor de așteptare și listelor.

Tema 7.

Algoritmi şi metode de căutare optimală a soluțiilor. Clasificarea algoritmilor. Avantaje şi dezavantaje ale algoritmilor din perspectiva: universalității, capacității de memorie utilizate și a vitezei de execuție.

CRITERII DE EVALUARE

Tema 1: *Tipul de date pointer. Alocare dinamică a memoriei. Funcții predefinite.*

1. **Utilizarea corectă a pointerilor în C:**
   - Demonstrarea înțelegerii conceptului de pointeri și utilizarea acestora în mod corespunzător în program.
   - Utilizarea pointerilor pentru accesarea, modificarea și transferul datelor între variabile.
2. **Alocarea dinamică a memoriei:**
   - Demonstrarea utilizării funcțiilor de alocare dinamică a memoriei, cum ar fi `malloc()`, `calloc()` sau `realloc()`.
   - Utilizarea corectă a alocării și eliberării memoriei pentru variabilele de tip pointer.
3. **Utilizarea funcțiilor predefinite de alocare dinamică a memoriei:**

- Utilizarea funcțiilor de alocare dinamică predefinite pentru nevoile specifice ale programului.
- Demonstrarea înțelegerii avantajelor și limitărilor fiecărei funcții (ex. `malloc(), calloc(), realloc()`).

4. **Manipularea datelor în tablouri 1D și 2D prin pointeri:**
   - Accesarea și manipularea elementelor tablourilor 1D și 2D utilizând pointeri.
   - Implementarea algoritmilor pentru operațiile comune cu tablourile, cum ar fi sortarea, căutarea, sau operații matriceale.
   - Demonstrația înțelegerii diferențelor în adresarea și manipularea datelor între tablouri unidimensionale și bidimensionale utilizând pointeri.
5. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului în limbajul C.
   - Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.

Tema 2: *Sortarea datelor. Algoritmi de sortare și analiza performanțelor. Tehnici și metode de sortare: 1) Bubble Sort; 2) Selection Sort; 3) Insertion Sort; 4) Merge Sort; 5) Quick Sort; 6) Shell Sort; 7) Counting Sort; 8) Comb Sort; 9) Radix Sort*.

1. **Implementarea corectă a algoritmilor de sortare:**
   - Implementarea corectă a algoritmilor specificați, inclusiv *Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, Shell Sort, Counting Sort, Comb Sort și Radix Sort*.
   - Asigurarea că algoritmii sunt implementați într-un mod care să sorteze corect datele în orice situație.
2. **Eficiența algoritmilor de sortare:**
   - Analiza corectitudinii și eficienței algoritmilor în contextul mărimii și structurii datelor de intrare.
   - Compararea și contrastarea performanței diferitor algoritmi de sortare în funcție de timpul de execuție și de spațiul necesar.
3. **Capacitatea de a analiza și de a explica performanța algoritmilor:**
   - Demonstrația înțelegerii principiilor de bază ale algoritmilor de sortare și a factorilor care influențează performanța acestora.
   - Capacitatea de a explica avantajele și dezavantajele fiecărui algoritm în diferite situații și contexte.
4. **Utilizarea corectă a tehnicilor și metodelor de sortare:**
   - Utilizarea corectă și eficientă a algoritmilor și tehnicilor de sortare specifice, inclusiv *Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, Shell Sort, Counting Sort, Comb Sort și Radix Sort*.
   - Selectarea și aplicarea tehnicilor și algoritmilor potriviți pentru tipul și dimensiunea datelor de intrare.
5. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului.
   - Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.

Tema 3: *Sortarea datelor. Heap Sort. Aplicarea dinamică arbori. Alocarea de memorie. Clasificarea arborilor dinamici. Arbori binari. Operații asupra arborilor binari.*

1. **Înțelegerea conceptelor de bază:**
   - Demonstrarea înțelegerii conceptelor fundamentale ale Heap Sort, aplicării dinamice a arborilor, alocării de memorie și clasificării arborilor dinamici.
   - Explicarea conceptelor cheie precum structura unui heap, procesul de heapify, operațiile de bază pe arbori binari etc.
2. **Implementarea corectă a Heap Sort:**
   - Implementarea corectă a algoritmului Heap Sort, evidențiind înțelegerea procedurii de construcție a unui heap și procesul de sortare folosind heap-ul construit.
   - Asigurarea că algoritmul funcționează corect pentru diverse seturi de date de intrare.
3. **Aplicarea dinamică a arborilor și alocarea de memorie:**
   - Demonstrarea utilizării aplicării dinamice a arborilor, inclusiv alocarea și eliberarea memoriei, pentru construirea și manipularea arborilor binari.
   - Utilizarea eficientă și corectă a funcțiilor de alocare dinamică a memoriei pentru a gestiona arborii binari.
4. **Clasificarea și operațiile asupra arborilor binari:**
   - Clasificarea corectă a arborilor binari în funcție de caracteristicile structurale și funcționale, cum ar fi arbori de căutare binari, arbori echilibrați, arbori rotaționali etc.
   - Implementarea corectă a operațiilor de bază pe arbori binari, cum ar fi inserția, ștergerea, căutarea, traversarea etc.
5. **Eficiența și performanța algoritmilor:**
   - Analiza și compararea eficienței și performanței algoritmului Heap Sort și a operațiilor pe arbori binari în diverse scenarii și pentru diferite dimensiuni ale datelor de intrare.
   - Demonstrarea înțelegerii avantajelor și limitărilor algoritmilor și operațiilor pe arbori binari în contextul diverselor cerințe de performanță și volum de date.
6. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului.
   - Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.

Tema 4: *Structuri statice de date struct, union, cu câmpuri de biţi. Declararea, inițializarea, citirea şi afișarea structurilor, aplicarea pointerilor. Structuri dinamice de date. Alocare de memorie. Elaborarea programelor în C, care operează cu diferite structure.*

1. **Înțelegerea conceptelor de bază:**
   - Demonstrația înțelegerii conceptelor de structuri statice de date, inclusiv structuri și uniuni, precum și a structurilor dinamice de date, cum ar fi alocarea dinamică a memoriei.
   - Explicarea corectă a conceptelor de declarare, inițializare, citire și afișare a structurilor, precum și aplicarea pointerilor în aceste operații.
2. **Implementarea corectă a operațiilor cu structuri:**
   - Implementarea corectă a operațiilor de citire, afișare și inițializare a structurilor statice și dinamice.
   - Demonstrarea înțelegerii modului de utilizare a pointerilor pentru accesarea și manipularea datelor din structuri.
3. **Utilizarea eficientă a alocării dinamice a memoriei:**

- Utilizarea corectă și eficientă a funcțiilor de alocare dinamică a memoriei, cum ar fi `malloc()`, `calloc()` sau `realloc()`, în contextul lucrului cu structuri dinamice de date.
- Asigurarea eliberării corecte a memoriei alocate pentru evitarea scurgerilor de memorie.

4. **Elaborarea programelor care operează cu diferite structuri:**
   - Elaborarea programelor care implică manipularea și procesarea datelor structurate utilizând atât structuri statice, cât și structuri dinamice de date.
   - Implementarea operațiilor specifice, cum ar fi sortarea, căutarea, inserarea, ștergerea etc., pe baza cerințelor specifice ale problemei.
5. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului în limbajul C.
   - Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.

Tema 5: *Tipul de date FILE. Fișiere de tip logic şi fizic. Funcții predefinite pentru date de tip FILE: fopen(), fread(), fwrite(), fprintf(), fscanf(), fseek(), feof(). Operații cu fișiere: creare, afișare, adăugare, corectare, sortare. Acces secvențial şi direct la fișiere.*

1. **Înțelegerea conceptelor de bază:**
   - Demonstrația înțelegerii conceptelor de fișiere de tip logic și fizic în limbajul C, precum și a tipului de date **FILE**.
   - Explicarea corectă a funcțiilor predefinite pentru lucrul cu fișiere: `fopen()`, `fread()`, `fwrite()`, `fprintf()`, `fscanf()`, `fseek()`, `feof()`.
2. **Implementarea corectă a operațiilor cu fișiere:**
   - Implementarea corectă a operațiilor de creare, afișare, adăugare, corectare și sortare a datelor din fișiere.
   - Demonstrarea înțelegerii și aplicarea funcțiilor predefinite pentru manipularea fișierelor în funcție de cerințele specifice ale problemei.
3. **Acces secvențial și direct la fișiere:**
   - Demonstrarea înțelegerii și aplicarea accesului secvențial și direct la fișiere utilizând funcțiile adecvate, cum ar fi `fseek()` pentru acces direct și `fread()` și `fwrite()` pentru acces secvențial.
   - Implementarea corectă și eficientă a operațiilor de citire și scriere în fișiere utilizând ambele metode de acces.
4. **Eficiența și performanța operațiilor cu fișiere:**
   - Analiza și compararea eficienței și performanței operațiilor cu fișiere în diverse scenarii și pentru diferite dimensiuni ale datelor.
   - Demonstrarea înțelegerii avantajelor și limitărilor utilizării accesului secvențial și direct la fișiere în contextul cerințelor specifice ale problemei.
5. **Gestionarea erorilor și manipularea excepțiilor:**
   - Tratarea și gestionarea corectă a erorilor în timpul operațiilor cu fișiere, cum ar fi erorile de deschidere sau de scriere/ citire.
   - Utilizarea adecvată a mecanismelor de tratare a excepțiilor pentru a asigura funcționarea corectă a programelor în condiții diverse.
6. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului în limbajul C.

- Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.

Tema 6: *Aplicații dinamice: stiva, şir de aşteptare, listă liniară, lista înlănțuită, listă bidirecțională. Operații asupra stivelor, șirurilor de așteptare și listelor.*

1. **Înțelegerea conceptelor de bază:**
   - Demonstrația înțelegerii conceptelor de stivă, șir de așteptare (queue) și liste (liniară, înlănțuită, bidirecțională) în cadrul programării dinamice.
   - Explicarea corectă a structurii și funcționalității fiecărei structuri de date.
2. **Implementarea corectă a operațiilor:**
   - Implementarea corectă a operațiilor specifice pentru stivă, șir de așteptare și liste, cum ar fi *push/pop* pentru stivă, *enqueue/dequeue* pentru șir de așteptare și operațiile specifice pentru listă (*inserție, ștergere, căutare etc*.).
   - Asigurarea că operațiile sunt implementate în mod eficient și că acestea respectă funcționalitatea specifică a fiecărei structuri de date.
3. **Utilizarea eficientă a memoriei și alocarea dinamică:**
   - Utilizarea eficientă a alocării dinamice a memoriei pentru gestionarea structurilor de date dinamice.
   - Asigurarea eliberării corecte a memoriei alocate pentru a evita scurgerile de memorie.
4. **Aplicarea corectă a structurilor de date în probleme practice:**
   - Aplicarea structurilor de date în rezolvarea problemelor practice, adaptându-le la cerințele specifice ale fiecărei probleme.
   - Demonstrația înțelegerii avantajelor și limitărilor fiecărei structuri de date în contextul problemelor rezolvate.
5. **Eficiența și performanța operațiilor:**
   - Analiza și compararea eficienței și performanței operațiilor pentru fiecare structură de date în diverse scenarii și pentru diferite dimensiuni ale datelor.
   - Demonstrarea înțelegerii avantajelor și limitărilor utilizării fiecărei structuri de date în contextul cerințelor specifice ale problemei.
6. **Codul și stilul de programare:**
   - Codul trebuie să fie bine structurat, ușor de înțeles și să respecte standardele de formatare a codului.
   - Utilizarea de comentarii explicative pentru a clarifica scopul și funcționalitatea blocurilor de cod.