Nama: _____ NRP: _____

LATIHAN SOAL STRUKTUR DATA

Bagian A. Isian Singkat (@ 2 poin)

1. Sebutkan 3 operasi dasar yang harus dimiliki oleh suatu struktur data tertentu __(a)__ , __(b)__, __(c)__
2. *Array* tersusun secara ___(a)___ di memori sehingga setiap elemennya dapat diakses secara ___(b)__.
3. Jenis Sorting yang memiliki kompleksitas O(n) jika memiliki *input* data yang sudah terurut adalah __(a)__, sementara *average case*-nya adalah O(_(b)_).
4. Salah satu metode *sorting* yang memiliki prinsip *comparison based* dengan *divide & conquer* adalah __(a)__, sementara salah satu metode *sorting* dengan prinsip *addres calculation* adalah __(b)__.
5. Mengapa operasi menyisipkan atau menghapus elemen pada bagian tengah *array* tidak efisien? Jelaskan secara singkat dalam satu kalimat.
6. Jika program Anda menyimpan sekumpulan data yang sering ditambah / dihapus namun jarang dibaca / diakses, struktur data linear apakah yang paling tepat untuk digunakan?
7. Pada *singly linked list* dengan *N* elemen (tidak memiliki *pointer* akhir), operasi pada bagian belakang *list* (hapus belakang atau tambah belakang) memiliki kompleksitas (dalam notasi big-O) _____.
8. Sistem *buffering* dapat diimplementasikan dengan ADT _____.
9. Jika ADT *stack* diimplementasikan dengan *singly linked list*, bagian *top* dari *stack* ditunjuk oleh *pointer* _____ dari *list*.
10. ADT *queue* dapat diimplementasikan (dengan *array*) secara _____ agar tidak perlu ada operasi penggeseran seluruh data ketika proses *dequeue*.
11. Suatu *queue* diimplementasikan dengan *doubly linked list* dengan `awal` merupakan bagian depan dari *queue*. Operasi *enqueue* akan sama dengan operasi __(a)__ pada *linked list*, sedangkan operasi *dequeue* akan sama dengan operasi ___(b)___ pada *linked list*.

12. Perhatikan potongan program berikut:

```
void f1(int m){
int i,j;
for(i=1;i<=k;i++)
   for(j=1;j<100;j++)
       x++;
}
```

Untuk nilai m yang sangat besar, berapakah kompleksitas potongan program tersebut O(__)

13. Diketahui kompleksitas dari 4 algoritme sebagai berikut:

    a. Algoritme A: $f_A(n) = n! + 1000$
    b. Algoritme B: $f_B(n) = \log n^2 + n^5$
    c. Algoritme C: : $f_C(n) = n^3 + 2^n$
    d. Algoritme D: $f_D(n) = n^3 \log n + n^3$

    Urutan algoritme yang memiliki eksekusi waktu yang paling lama hingga tercepat ialah _____

Nama: _____ NRP: _____

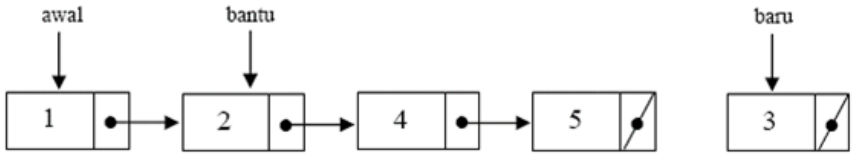
14. Sebutkan kondisi *worst case* pada kasus *sequential search* dalam 1 kalimat singkat serta kompleksitasnya dalam notasi big-O.
15. *Best case* pada kasus pencarian nilai minimal dari bilangan bulat sebanyak *m* yang tersimpan pada suatu array tidak terurut adalah O(___).

Bagian B. Uraian

1. Berikut ini merupakan implementasi *stack* yang berisi *integer* dengan menggunakan *array*. Lengkapilah fungsi `push()` dan `pop()`. Tidak perlu membuat fungsi `main()`.

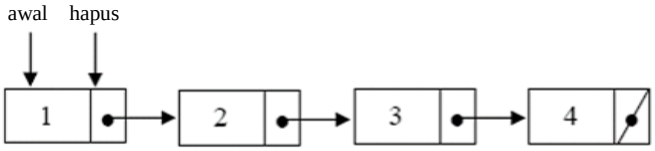
```
struct stack {
    int *data;
    int kapasitas;
    int top;
};

void inisialisasi(stack* S, int n) {
    S->top = -1;
    S->kapasitas = n;
    S->data = malloc(n * sizeof(int));
}

void push(stack* S, int data) {
    if( /* pengecekan overflow */ ) {
        cout << "Stack overflow" << endl;
    } else {
        /* lakukan push */
        /* kode boleh dalam beberapa baris */
    }
}

int pop(stack* S) {
    if( /* pengecekan underflow */ ) {
        cout << "Stack underflow" << endl;
        return 0;
    } else {
        /* lakukan pop dan kembalikan nilai */
        /* kode boleh dalam beberapa baris */
    }
}
```

2. **(20 poin)** Perhatikan ilustrasi dua operasi pada *singly linked list* berikut. Buatlah algoritme untuk setiap operasi (*pseudocode*, tidak perlu membuat program lengkap). Nama variabel harus sesuai dengan gambar (misalnya `awal`, `bantu`, `baru`, `hapus`).
   a. Menyisipkan data di tengah (setelah *pointer* `bantu`).



   b. Menghapus data di akhir. Kondisi permulaan digambarkan pada diagram berikut.

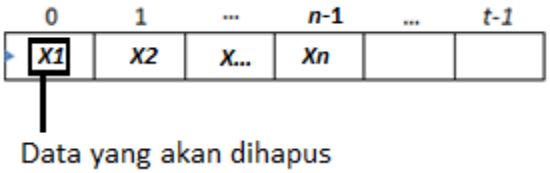
Nama: _____ NRP: _____



3. **(20 poin)** Bandingkanlah proses penghapusan elemen pertama pada struktur data Array dan *doubly linked list*.

   a) Buatlah algoritme untuk proses tersebut (*pseudocode*, tidak perlu membuat program lengkap) jika struktur data yang digunakan ialah

      i. Array, dengan asumsi array tidak pernah kosong, array selalu terisi dari indeks ke-0. Sebagai petunjuk gunakan ilustrasi berikut:



Keterangan:

- *n* adalah jumlah data sebelum penghapusan
- *t* adalah kapasitas maksimum array

      ii. Doubly Linked List, sebagai petunjuk gunakan ilustrasi berikut:

   b) Tentukan kompleksitas dalam notasi Big-Oh untuk masing-masing algoritme tersebut!

   c) Algoritme mana yang lebih cepat beserta alasannya!

4. **(10 poin)** Terdapat urutan data sebagai berikut: 398, 532, 697, 244, 903, 157, 165, 212, 449, 790
   Lakukan pengurutan pada data tersebut dengan teknik **Radix Sort!**