Nama : Li Wei

Program studi/Kelas : Sistem Informasi/B

Mata Kuliah : Algoritms and Programming

Latihan soal bab 3

8. a. Buat fungsi untuk menghitung banyaknya huruf hidup (vokal) dalam sebuah string.
   b. Buat fungsi untuk menghitung banyaknya huruf mati (konsonan) dalam sebuah string.
   c. Dengan menggunakan fungsi pada 8.a dan 8.b buatlah fungsi untuk menghitung banyaknya karakter selain huruf yang ada dalam sebuah string.
   d. Buat pula algoritma utamanya.
   Jawab:
   Algoritma fungsi_menentukan_jenis_karakter
   Algoritma ini akan menentukan jenis karakter (huruf vocal, konsonan, dan nonhuruf) dan jumlahnya pada kalimat yang kita input, variable x bertipe string, variable vocal, konsonan, karakter bersifat integer.

```
[Deklarasi Fungsi]
VOID huruf_vokal (string x, int vokal){
        for(int i = 0; i < x.length(); i++){
        if(x[i]=='a'||x[i]=='i'||x[i]=='u'||x[i]=='e'||x[i]=='o'||
                x[i]=='A'||x[i]=='I'||x[i]=='U'||x[i]=='E'||x[i]=='O')
        vokal++;
                }
                if (vokal > 0) {
        write  ( "Jumlah huruf vokal = "  vocal)
                }
                else {
        write ( "Huruf vokal tidak ditemukan")
        }
}

VOID huruf_konsonan (string x, int konsonan){
        for(int i = 0; i < x.length(); i++){
        if(x[i]=='b'||x[i]=='c'||x[i]=='d'||x[i]=='f'||x[i]=='g'||x[i]=='h'||
                x[i]=='j'||x[i]=='k'||x[i]=='l'||x[i]=='m'||x[i]=='n'||x[i]=='p'||x[i]=='q'||

        x[i]=='r'||x[i]=='s'||x[i]=='t'||x[i]=='v'||x[i]=='w'||x[i]=='x'||x[i]=='y'||x[i]=='z'||
                x[i]=='B'||x[i]=='C'||x[i]=='D'||x[i]=='F'||x[i]=='G'||x[i]=='H'||
                x[i]=='J'||x[i]=='K'||x[i]=='L'||x[i]=='M'||x[i]=='N'||x[i]=='P'||x[i]=='Q'||

        x[i]=='R'||x[i]=='S'||x[i]=='T'||x[i]=='V'||x[i]=='W'||x[i]=='X'||x[i]=='Y'||x[i]=='Z')
        konsonan++;
                }
                if (konsonan > 0) {
```

```
                write  ( "Jumlah huruf konsonan = "  konsonan)
                }
                else {
                write ( "Huruf konsonan tidak ditemukan")
        }
}

VOID huruf_karakter (string x, int karakter){
        for(int i = 0; i < x.length(); i++){
        if(x[i]=='`'||x[i]=='~'||x[i]=='!'||x[i]=='1'||x[i]=='@'||x[i]=='2'||
                x[i]=='#'||x[i]=='3'||x[i]=='$'||x[i]=='4'||x[i]=='%'||x[i]=='5'||x[i]=='^'||

        x[i]=='6'||x[i]=='&'||x[i]=='7'||x[i]=='*'||x[i]=='8'||x[i]=='('||x[i]=='9'||x[i]==')'||
                x[i]=='0'||x[i]=='-'||x[i]=='_'||x[i]=='='||x[i]=='+'||x[i]=='|'||
                x[i]==';'||x[i]==':'||x[i]=='"'||x[i]=='<'||x[i]==','||x[i]=='>'||x[i]=='.'||
                x[i]=='/'||x[i]=='?')
        karakter++;
                }
                if (karakter > 0) {
                write ( "Jumlah huruf karakter = "  karakter)
                }
                else {
                write ( "Huruf karakter tidak ditemukan")
        }
}

1. [Algoritma Utama]
[Deklarasi Variabel]
string x
int vokal = 0
int konsonan = 0
int karakter = 0

2. write ( "Input kata / kalimat: ")
   getline(read (x))

3. [Pemanggilan fungsi]
   huruf_vokal (x, vokal);
   huruf_konsonan(x, konsonan);
   huruf_karakter(x, karakter);

4. [Selesai]
   Halt
```

9. a. Buat fungsi REVERSE (iteratif dan rekursif) untuk menerima string input dan menghasilkan string output yang isinya adalah kebalikan urutan karakter dalam string input.

Sebagai contoh string input = “abcd”, maka string output = “dcba”.Buat pula algoritma utamanya.
b. Buat pula algoritma utamanya.
Jawab:
Algoritma membalik_kata_reverse
Algoritma untuk membuat kata yang diinput menjadi terbalik dengan fungsi dan string. Variable kata merupakan string, sisa variablenya merupakan integer.

```
[Deklarasi Fungsi Iteratif]
VOID reverse_iteratif (int x, string kata){
write ("Jika menggunakan fungsi iteratif: ")
for (int a = x-1; a >= 0; a--){
write ( kata[a])
   }
}

[Deklarasi Fungsi Rekursif]
VOID reverse_rekursif (int x, int y, string &kata){
if (y > (x-1-y)){
return
}
swap (kata[y], kata[x-y-1]);
y++
reverse_rekursif (x, y, kata)
}

1. [Algoritma Utama]
[Deklarasi Variabel]
string kata
int y = 0
2. [Masukkan kata yang ingin direverse/dibalik]
Write ( "Masukkan kata yang ingin dibalik: ")
getline (read (kata))
int    x = kata.length()
3. [mendeklarasi Fungsi]
reverse_iteratif(x, kata)
reverse_rekursif (x, y, kata)
write ("dengan rekursif: " kata)
```

10. a. Palindrome adalah string yang dapat dibaca dari depan maupun dari belakang. Contohnya adalah ”malam”, ”kodok” dan lain-lain. Buatlah fungsi (rekursif dan iteratif) untuk menentukan apakah sebuah string input termasuk palindron atau bukan.Diasumsikan string input berupa huruf kecil semua. Buat pula algoritma utamanya.
    b. Buat pula algoritma utamanya.
    Jawab:
    Algoritma menentukan_palindrome

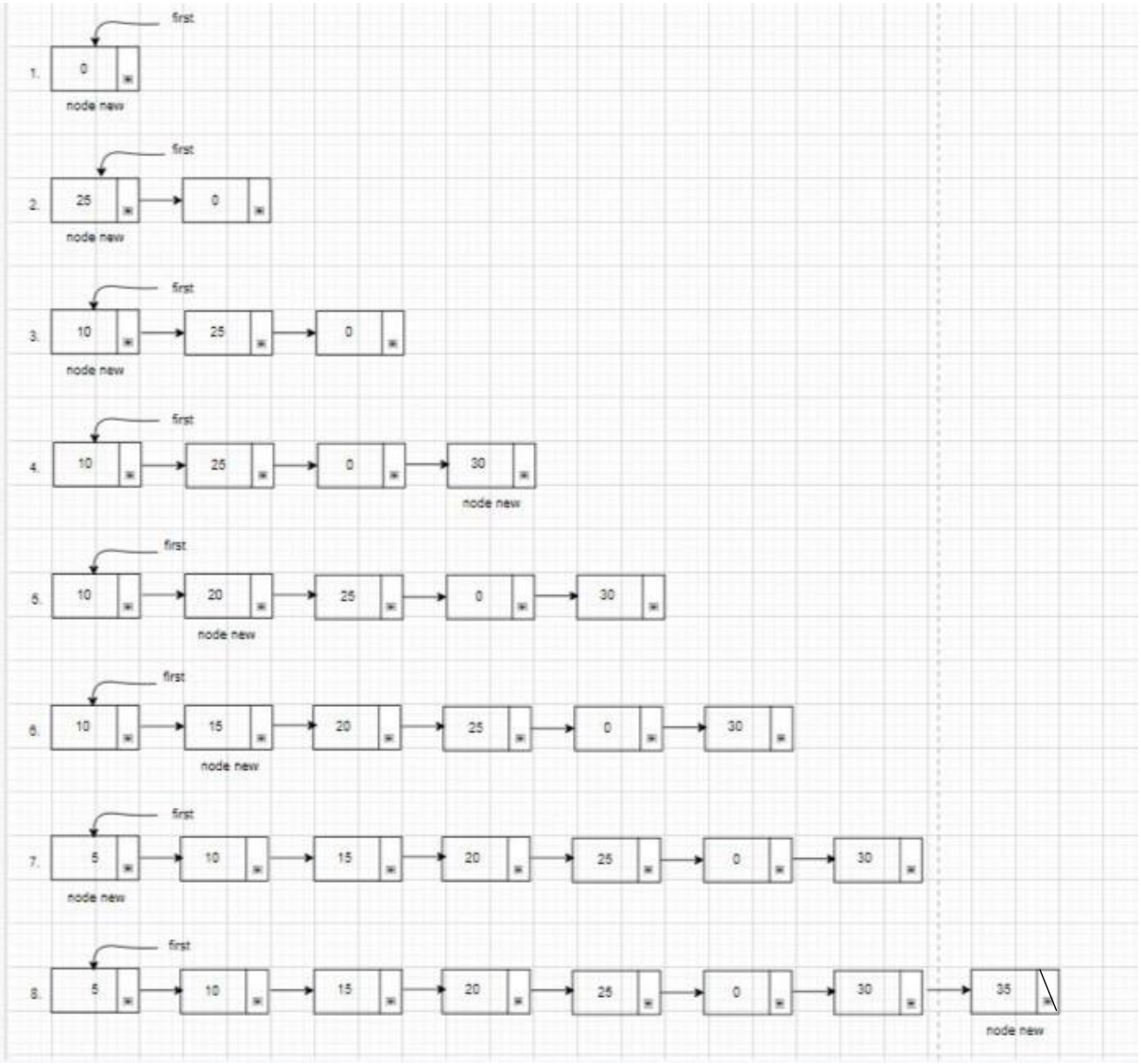
Algoritma ini akan menentukan kata yang kita input untuk menentukan apakah kata tersebut merupakan kata palindrome. Variable kata merupakan string, sisa variabelnya merupakan integer.

```
[Deklarasi Fungsi Iteratif]
INT palindrome_iteratif (string kata){
   INT x, i
   x = kata.length()
   for (i = 0; i < x; i++){
      if (kata[i] != kata[x-i-1]){
          Write ( "Kata "  kata  " bukanlah kata palindrome " )
          return 0
      }
   }
   Write ( "Kata "  kata  " merupakan kata palindrome ")
   return true
}

[Deklarasi Fungsi Rekursif]
INT palindrome_rekursif(char str[],int s, int e){
if (s == e)
   return true
if (str[s] != str[e])
   return false
if (s < e + 1)
   return palindrome_rekursif (str, s + 1, e - 1)
   return true
}
INT palindrome(char str[])
{
   int n = strlen(str)
    if (n == 0)
      return true

   return palindrome_rekursif(str, 0, n - 1)
}
1. [Algoritma Utama]
[Deklarasi Variabel]
string kata
2. [Masukkan kata yang inginkan]
write ( "Input Kata : ")
read (kata)
3. [Deklarasi fungsi]
write ((palindrome_iteratif) (kata))
char string1[20];
write ( "Masukkan kata: ")
read ( string1)
4. [Menentukan kata tersebut palindrome atau bukan]
```

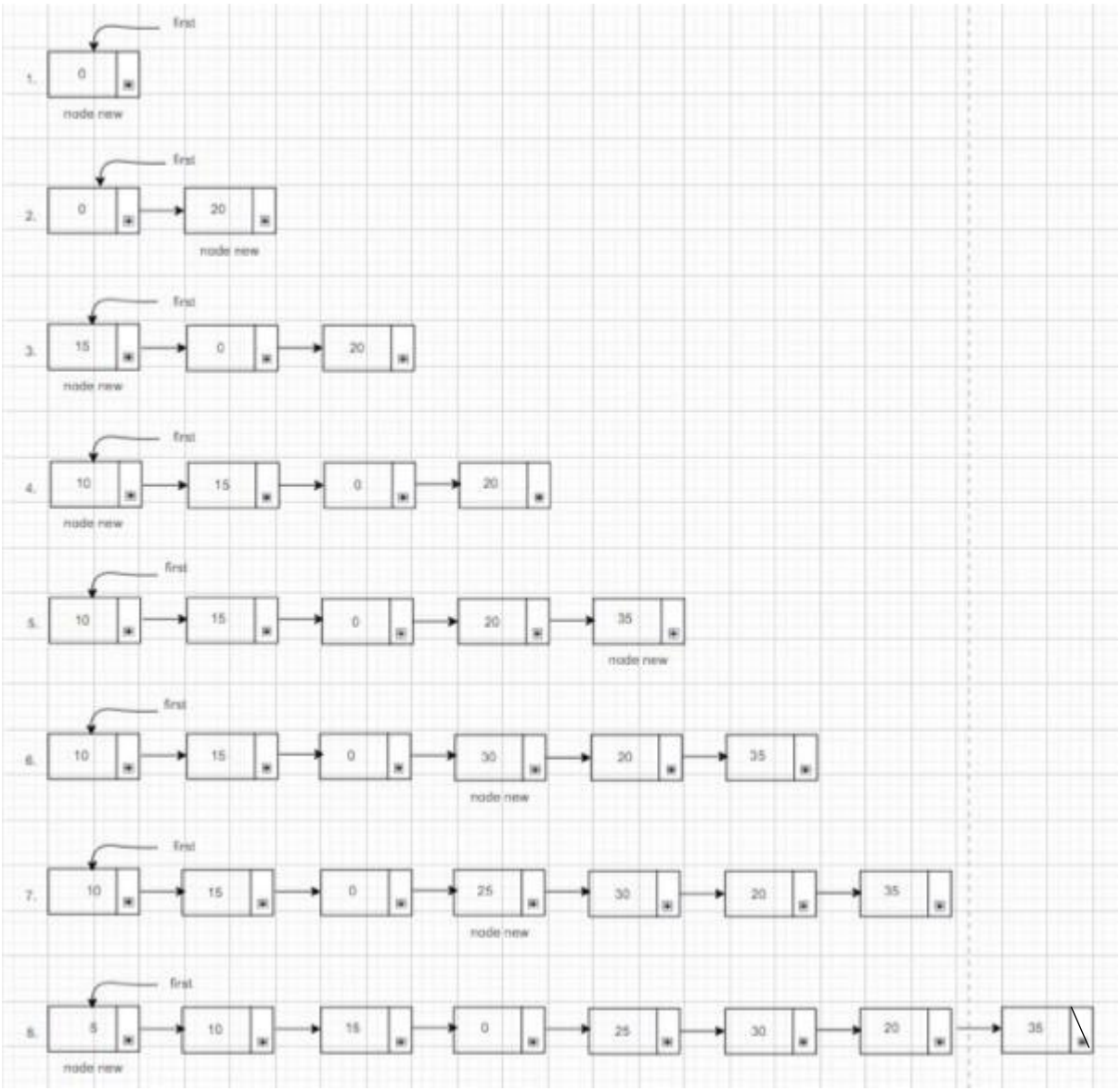
if (palindrome(string1))
write ("Kata " string1 " merupakan kata palindrome")
else
write ("Kata l" string1 " bukan kata palindrome")
5. [Selesai]
Halt

Latihan soal bab 5.5

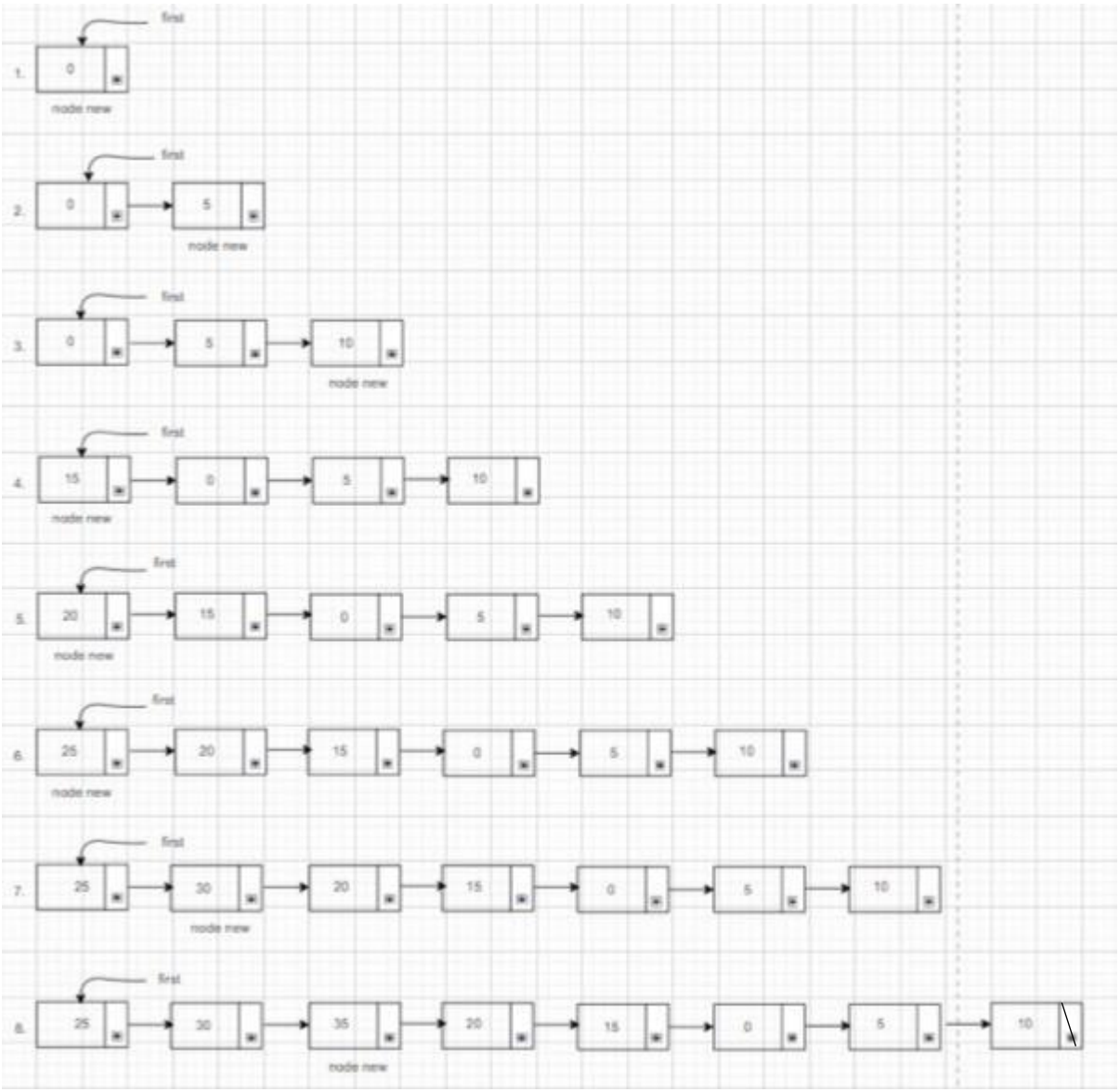
1. a.



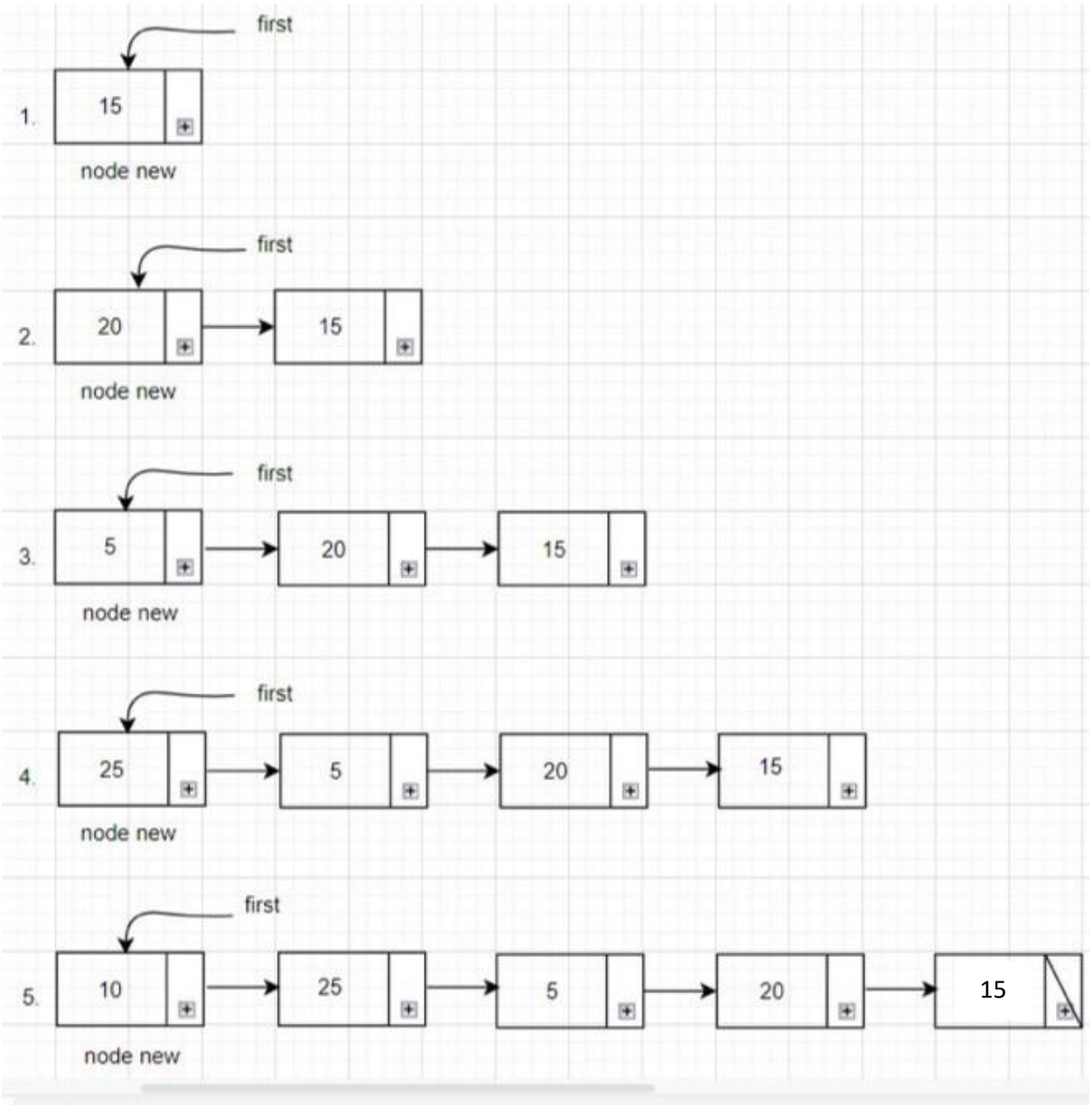
1. b.


1.
first
0
node new
2.
first
0
20
node new
3.
first
15
0
20
node new
4.
first
10
15
0
20
node new
5.
first
10
15
0
20
35
node new
6.
first
10
15
0
30
20
35
node new
7.
first
10
15
0
25
30
20
35
node new
8.
first
5
10
15
0
25
30
20
35
node new

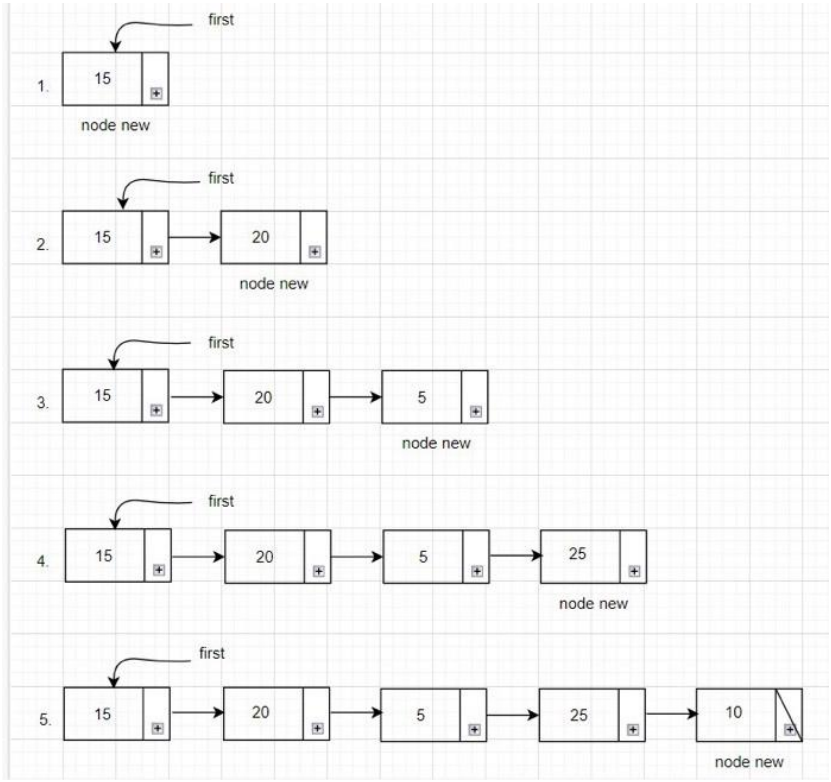
1. c.


first
1.
0
node new
first
2.
0
5
node new
first
3.
0
5
10
node new
first
4.
15
0
5
10
node new
first
5.
20
15
0
5
10
node new
first
6.
25
20
15
0
5
10
node new
first
7.
25
30
20
15
0
5
10
node new
first
8.
25
30
35
20
15
0
5
10
node new

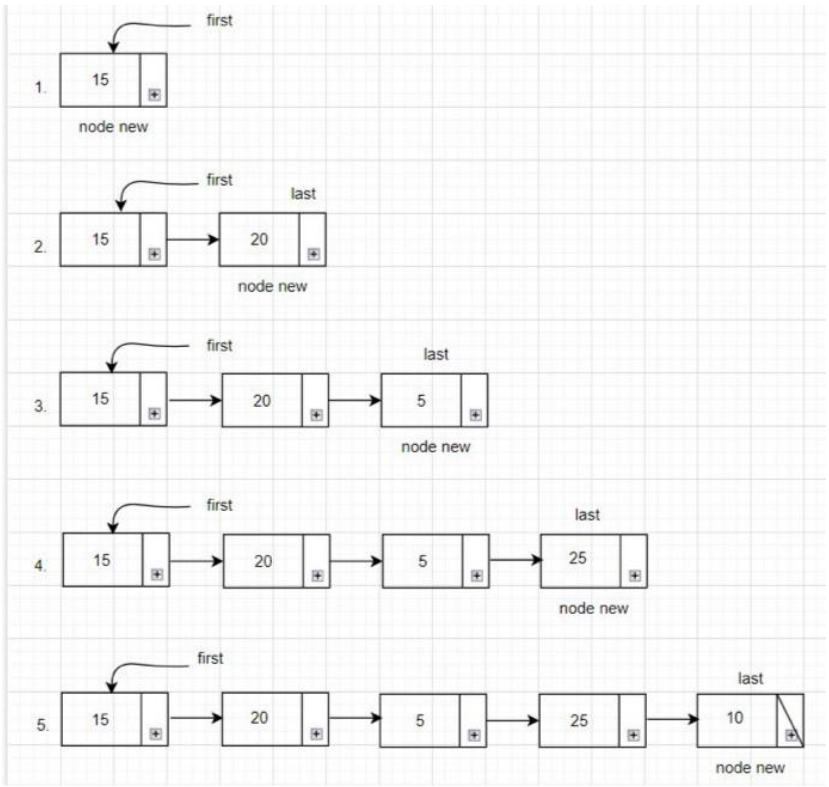
2. a.


first
1.
15
node new
first
2.
20
15
node new
first
3.
5
20
15
node new
first
4.
25
5
20
15
node new
first
5.
10
25
5
20
15
node new

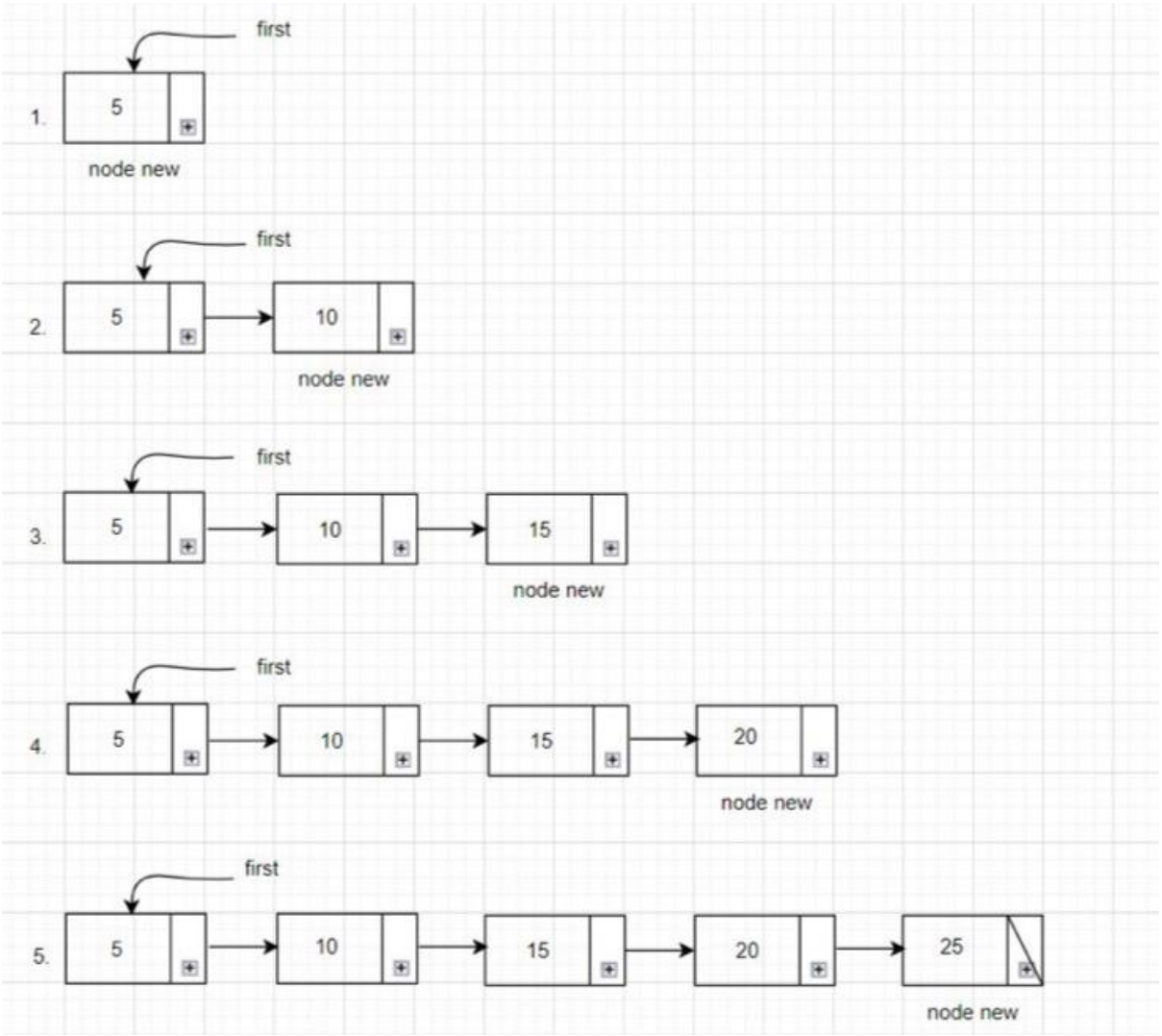
2. b.


first
1.
15
node new
first
2.
15
20
node new
first
3.
15
20
5
node new
first
4.
15
20
5
25
node new
first
5.
15
20
5
25
10
node new

2. c.


first
1.
15
node new
first
last
2.
15
20
node new
first
last
3.
15
20
5
node new
first
last
4.
15
20
5
25
node new
first
last
5.
15
20
5
25
10
node new

2. d.



3. int hitung_node (node* first){

```
 node* temp = first;
 int a = 0;
 while (temp != NULL){
  a++;
  temp = temp -> next;
 }
 return a;
}
```

4. #include <bits/stdc++.h>

```
using namespace std;

struct Node {
   int data;
   Node* next;
};

//FUNGSI UNTUK MEMBUAT DAN MENGEMBALIKAN HASIL
Node* getNode(int data)
{
   // allocating space
   Node* newNode = (Node*)malloc(sizeof(Node));

   // inserting the required data
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

//FUNGSI UNTUK INSERT MIDDILE DI DALAM LINKED LIST
void insertAtMid(Node** head_ref, int x)
{
   //APABILA LINK KOSONG
   if (*head_ref == NULL)
      *head_ref = getNode(x);
   else {

      // MENDAPATKAN NODE BARU
      Node* newNode = getNode(x);
```

```
        Node* ptr = *head_ref;
        int len = 0;

        // calculate length of the linked list
        //, i.e, the number of nodes
        while (ptr != NULL) {
            len++;
            ptr = ptr->next;
        }

        // 'count' the number of nodes after which
        //  the new node is to be inserted
        int count = ((len % 2) == 0) ? (len / 2) : (len + 1) / 2;
        ptr = *head_ref;

        // 'ptr' points to the node after which
        // the new node is to be inserted
        while (count-- > 1)
            ptr = ptr->next;

        // insert the 'newNode' and adjust the
        // required links
        newNode->next = ptr->next;
        ptr->next = newNode;
    }
}

// function to display the linked list
void display(Node* head)
{
    while (head != NULL) {
```

```
        cout << head->data << " ";
        head = head->next;
    }
}

// Driver program to test above
int main()
{
            int a, b, c, d, e;
    // Creating the list 1->2->4->5
    Node* head = NULL;
    cout << "masukkan elemen pertama = ";
    cin >> a;
    cout << "masukkan elemen kedua = ";
    cin >> b;
    cout << "masukkan elemen keempat = ";
    cin >> d;
    cout << "masukkan elemen kelima = ";
    cin >> e;

    head = getNode(a);
    head->next = getNode(b);
    head->next->next = getNode(d);
    head->next->next->next = getNode(e);
    cout << "Linked list before insertion: ";
    display(head);
    cout << endl;

    cout << "masukkan elemen tengah = ";
    cin >> c;
```

```
    insertAtMid(&head, c);

    cout << "\nLinked list after insertion: ";
    display(head);

    return 0;
}
```

5.
```
#include <bits/stdc++.h>
using namespace std;

class node {
public:
    int data;
    node* next;

    // A constructor is called here
    node(int value)
    {

        // It automatically assigns the value to the data
        data = value;

        // Next pointer is pointed to NULL
        next = NULL;
    }
};

// Function to insert an element at head position
void insertathead(node*& head, int val)
{
```

```
    node* n = new node(val);
    n->next = head;
    head = n;
}

// Function to insert an element at the end
void insertattail(node*& head, int val)
{
    node* n = new node(val);
    if (head == NULL) {
        head = n;
        return;
    }

    node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = n;
}

// Function to print the singly linked list
void print(node*& head)
{
    node* temp = head;

    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
```

```
}

// Main function
int main()
{

   // Declaring an empty linked list
   node* head = NULL;

   insertathead(head, 10);
   insertathead(head, 5);
   cout << "After insertion at head: ";
   print(head);
   cout << endl;

         insertattail(head, 15);
   insertattail(head, 20);
   insertattail(head, 25);
   cout << "After insertion at tail: ";
   print(head);
   cout << endl;

   return 0;
}

6. #include <bits/stdc++.h>
using namespace std;

// A linked list node
class Node{
         public:
```

```
        int data;
        Node *next;
};

/* Given a reference (pointer to pointer)
to the head of a list and an int, inserts
a new node on the front of the list. */
void push(Node** head_ref, int new_data){ //function to do insert front, to push new node in front of the main node/null
        // 1. Allocate node
        Node* new_node = new Node();

        // 2. Put in the data
        new_node->data = new_data;

        // 3. Make next of new node as head
        new_node->next = (*head_ref);

        // 4. Move the head to point to the
        // new node
        (*head_ref) = new_node;
}

void printList(Node *node){ //function to show the final form of the node
        while (node != NULL)
        {
                cout<<" "<<node->data;
                node = node->next;
        }
}
```

```
// Driver code
int main()
{
	// Start with the empty list
	Node* head = NULL;

	// Insert 25. So linked list
	// becomes 25->NULL
	push(&head, 25);

	// Insert 20 at the beginning/as front.
	// So linked list becomes
	// 20->25->NULL
	push(&head, 20);

	// Insert 15 at the beginning/as front.
	// So linked list becomes
	// 15->20->25->NULL
	push(&head, 15);

	// Insert 10 at the beginning/as front.
	// So linked list becomes
	// 10->15->20->25->NULL
	push(&head, 10);

	// Insert 5 at the beginning/as front.
	// So linked list becomes
	// 5->10->15->20->25->NULL
	push(&head, 5);

	cout << "Created Linked list by insert front are: ";
```

```
        printList(head);


        return 0;
}
```

7.
```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *next;
} *temp = NULL, *first = NULL, *second = NULL, *third = NULL, *last = NULL;
struct Node* Create (int A[], int n)
{
   int i;
   struct Node *t, *last;
   temp = (struct Node *) malloc(sizeof(struct Node));
   temp->data = A[0];
   temp->next = NULL;
   last = temp;
   for (i = 1; i < n; i++)
   {
      t = (struct Node *) malloc(sizeof(struct Node));
      t->data = A[i];
      t->next = NULL;
      last->next = t;
      last = t;
   }
   return temp;
}
void Display(struct Node *p)
```

```
{
   while (p != NULL)
   {
      printf ("%d ", p->data);
      p = p->next;
   }
}
void Merge(struct Node *first, struct Node *second)
{
   if (first->data < second->data)
   {
      third = last = first;
      first = first->next;
      last->next = NULL;
   }
   else
   {
      third = last = second;
      second = second->next;
      last->next = NULL;
   }

   while (first != NULL && second != NULL)
   {
      if (first->data < second->data)
      {
         last->next = first;
         last = first;
         first = first->next;
         last->next = NULL;
      }
```

```
        else
        {
            last->next = second;
            last = second;
            second = second->next;
            last->next = NULL;
        }
    }

    if (first != NULL)
        last->next = first;
    else
        last->next = second;
}
int main()
{
    int A[] = { 15, 20, 5 };
    int B[] = { 25,10 };
    first = Create (A, 4);
    second = Create (B, 4);
    printf ("1st Linked List: ");
    Display (first);
    printf ("\n2nd Linked List: ");
    Display (second);
    Merge (first, second);
    printf ("\n\nMerged Linked List: \n");
    Display (third);
  return 0;
}
```

8a.
```
#include <bits/stdc++.h>
using namespace std;

class Node
{
   public:
   int data;
   Node *next;
};

//fungsi split linkedlist menjadi 2
void splitList(Node *head, Node **head1_ref,
                      Node **head2_ref)
{
   Node *second_ptr = head;
   Node *first_ptr = head;

   if(head == NULL)
      return;

   /* If there are odd nodes in the circular list then
     fast_ptr->next becomes head and for even nodes
     fast_ptr->next->next becomes head */
   while(first_ptr->next != head &&
        first_ptr->next->next != head)
   {
      first_ptr = first_ptr->next;
      second_ptr = second_ptr->next->next;
   }

   /* If there are even elements in list
```

```
      then move fast_ptr */
    if(first_ptr->next->next == head)
       first_ptr = first_ptr->next;

    /* Set the head pointer of first half */
    *head1_ref = head;

    /* Set the head pointer of second half */
    if(head->next != head)
       *head2_ref = second_ptr->next;

    /* Make second half circular */
    first_ptr->next = second_ptr->next;

    /* Make first half circular */
    second_ptr->next = head;
}

//push function
void push(Node **head_ref, int data)
{
    Node *ptr1 = new Node();
    Node *temp = *head_ref;
    ptr1->data = data;
    ptr1->next = *head_ref;

    /* If linked list is not NULL then
      set the next of last node */
    if(*head_ref != NULL)
    {
       while(temp->next != *head_ref)
```

```
        temp = temp->next;
        temp->next = ptr1;
    }
    else
        ptr1->next = ptr1; /*For the first node */

    *head_ref = ptr1;
}

//print linked list
void printList(Node *head)
{
    Node *temp = head;
    if(head != NULL)
    {
        cout << endl;
        do {
        cout << temp->data%2 << " ";
        temp = temp->next;
        } while(temp != head);
    }
}

int main()
{
    int list_size, i;

    Node *head = NULL;
    Node *head1 = NULL;
    Node *head2 = NULL;
```

```
    push(&head, 15);
    push(&head, 20);
    push(&head, 5);
    push(&head, 25);

    cout << "Linked List AWAL :";
    printList(head);

    //split linked list menjadi 2
    splitList(head, &head1, &head2);

    cout << endl;
    cout << "Linked List Pointer FIRST : ";
    printList(head1);

            cout << endl;
    cout << "Linked List Pointer SECOND : ";
    printList(head2);
    return 0;
}
```

8b. void yg dipake = movenode dan pisahlink

sisanya cuma yg kek output linked list dkknya

```
#include <iostream>
using namespace std;

struct node {
 int data;
 node* next;
};
```

```
//output pointer
void printlist (node*n){
 while (n!=NULL){
  cout << n -> data << endl;
  n = n -> next; //membuat n menunjuk pada node berikutnya
 }
}

//menghitung node ada berapa (NO 3)
int hitung_node (node* first){
 node* temp = first;
 int a = 0;
 while (temp != NULL){
  a++;
  temp = temp -> next;
 }
 return a;
}

void movenode(struct node** destRef, struct node** sourceRef)
{
   // if the source list empty, do nothing
   if (*sourceRef == NULL) {
      return;
   }

   node* newnode = *sourceRef;  // the front source node
   *sourceRef = (*sourceRef)->next;    // advance the source pointer
   newnode->next = *destRef;          // link the old dest off the new node
   *destRef = newnode;                // move dest to point to the new node
```

```
}
void pisahlink(node* source, node** aRef, node** bRef)
{
   // Split the nodes into a and b lists
   struct node* a = NULL;
   struct node* b = NULL;

   struct node* current = source;

   while (current != NULL)
   {
      movenode(&a, &current);        // Move a node to a

      if (current != NULL) {
         movenode(&b, &current);     // Move a node to b
      }
   }

   *aRef = a;
   *bRef = b;
}

//ALGORITMA UTAMA
int main (){
 int x;
 node linkedlist;

 node* onee = NULL;
 node* twoo = NULL;
 node* first = new node ();
 node* second = new node();
```

```
node* third = new node ();
node* fourth = new node ();
node* fifth = new node();

first -> data = 5;
first -> next = second;
second -> data = 10;
second -> next = third;
third -> data = 15;
third -> next = fourth;
fourth -> data = 20;
fourth -> next = fifth;
fifth -> data = 25;
fifth -> next = NULL;

printlist (first);
cout << endl;
cout << "Jumlah node linked list sebanyak: " << hitung_node(first);
cout << endl;
pisahlink (first, &onee, &twoo);
cout << "Split pertama: ";
printlist (onee);
cout << endl;
cout << endl << "Split kedua: ";
printlist (twoo);

return 0;

}
```

9. #include <bits/stdc++.h>

```
using namespace std;



struct Node {
    int data;
    struct Node* next;
    Node(int data)
    {
        this->data = data;
        next = NULL;
    }
};

struct LinkedList {
    Node* head;
    LinkedList()
    {
        head = NULL;
    }

    Node* reverse(Node* head)
    {
        if (head == NULL || head->next == NULL)
            return head;
        // Memanggil rekursif
        Node* rest = reverse(head->next);
        head->next->next = head;

        head->next = NULL;

        return rest;
```

```
    }

    void print()
    {
        struct Node* temp = head;
        while (temp != NULL) {
            cout << temp->data << " ";
            temp = temp->next;
        }
    }

    void push(int data)
    {
        Node* temp = new Node(data);
        temp->next = head;
        head = temp;
    }
};

int main()
{
    LinkedList ll;
    ll.push(15);
    ll.push(20);
    ll.push(5);
    ll.push(25);
    ll.push(10);

    cout << "Linked List sebelum :\n";
    ll.print();
```

```
    ll.head = ll.reverse(ll.head);

    cout << "\nLinked List sesudah :\n";
    ll.print();
    return 0;
}
```

10.
```
#include <bits/stdc++.h>
using namespace std;

struct tnode {
        int nilai;
        tnode *next;
};

tnode *awal = NULL, *akhir = NULL, *bantu;
int dat, y=1, s, d;

void baca (int x) {
        tnode *baru;
        baru = new tnode;
        cout << " Masukan Nilai : ";
        cin  >> baru->nilai;
        cout << endl;

        baru->next = NULL;
        awal = baru;
        akhir = baru;
        akhir->next = NULL;

        for (y; y<x; y++) {
```

```
                baru = new tnode;
                cout << " Masukan Nilai : ";
                cin  >> baru->nilai;
                cout << endl;

//              baru->next = NULL;
                baru->next = awal;
                awal = baru;
        }
}

void cetak () {
        bantu = awal;
        cout << endl;
        while (bantu != NULL) {
                cout << bantu->nilai << "  ";
                bantu = bantu->next;
        }
        cout << endl;
}

void hapus(int M, int N)
{
   tnode *curr = awal, *t;
   int count;

   while (curr) {
      for (count = 1; count < M && curr!= NULL; count++)
         curr = curr->next;

      if (curr == NULL)
```

```
            return;

        t = curr->next;
        for (M; M<=N && t!= NULL; M++) {
            tnode *temp = t;
            t = t->next;
            free(temp);
        }

        curr->next = t;
        curr = t;
    }
}


int main () {
            tnode* awal = NULL;

            cout << " Banyaknya Data : ";
            cin >> dat;
            cout << endl;

            baca (dat);
            cout << "Data Link List : ";
            cetak ();
            cout << endl;
            cout << "Urutan Link List yang akan dihapus i sampai j (cnth:1 2): ";
            cin >> s;
            cin >> d;

            hapus (s, d); // (M, N)
```

```
	cout << endl;
	cout << "Data Link List Setelah dihapus : ";
	cetak ();

	return 0;
}
```