Nama : Ballogom Sidabutar

NIM : 103112400131

Kelas : 12-IF-04

Soal Nomor 1 (Single Linked List)

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

// ==================== DEFINISI STRUCT ====================

struct Product {
   string Nama;
   string SKU;
   int Jumlah;
   float HargaSatuan;
   float DiskonPersen;
};

struct Node;
typedef Node* address;

struct Node {
   Product info;
   address next;
};

struct List {
   address head;
};

// ==================== DEKLARASI FUNGSI/PROSEDUR ====================

float hitungHargaAkhir(Product P);
bool isEmpty(List L);
void createList(List &L);
address allocate(Product P);
void deallocate(address addr);
void insertFirst(List &L, Product P);
void insertLast(List &L, Product P);
void insertAfter(List &L, address Q, Product P);
void deleteFirst(List &L, Product &P);
void deleteLast(List &L, Product &P);
void deleteAfter(List &L, address Q, Product &P);
void updateAtPosition(List &L, int posisi, Product P);
void viewList(List L);
void searchByFinalPriceRange(List L, float minPrice, float maxPrice);
void MaxHargaAkhir(List L);

// ==================== IMPLEMENTASI FUNGSI/PROSEDUR ====================

float hitungHargaAkhir(Product P) {
   return P.HargaSatuan * (1 - P.DiskonPersen / 100.0);
}

bool isEmpty(List L) {
   return L.head == NULL;
}
```

```
void createList(List &L) {
   L.head = NULL;
}

address allocate(Product P) {
   address newNode = new Node;
   newNode->info = P;
   newNode->next = NULL;
   return newNode;
}

void deallocate(address addr) {
   delete addr;
}

void insertFirst(List &L, Product P) {
   address newNode = allocate(P);
   if (isEmpty(L)) {
      L.head = newNode;
   } else {
      newNode->next = L.head;
      L.head = newNode;
   }
}

void insertLast(List &L, Product P) {
   address newNode = allocate(P);
   if (isEmpty(L)) {
      L.head = newNode;
   } else {
      address temp = L.head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

void insertAfter(List &L, address Q, Product P) {
   if (Q == NULL) {
      cout << "Node Q tidak valid!" << endl;
      return;
   }
   address newNode = allocate(P);
   newNode->next = Q->next;
   Q->next = newNode;
}

void deleteFirst(List &L, Product &P) {
   if (isEmpty(L)) {
      cout << "List kosong, tidak ada yang bisa dihapus!" << endl;
      return;
   }
   address temp = L.head;
   P = temp->info;
   L.head = L.head->next;
   deallocate(temp);
}

void deleteLast(List &L, Product &P) {
   if (isEmpty(L)) {
      cout << "List kosong, tidak ada yang bisa dihapus!" << endl;
```

```
    return;
  }

  if (L.head->next == NULL) {
    P = L.head->info;
    deallocate(L.head);
    L.head = NULL;
  } else {
    address temp = L.head;
    while (temp->next->next != NULL) {
      temp = temp->next;
    }
    P = temp->next->info;
    deallocate(temp->next);
    temp->next = NULL;
  }
}

void deleteAfter(List &L, address Q, Product &P) {
  if (Q == NULL || Q->next == NULL) {
    cout << "Tidak ada node setelah Q atau Q tidak valid!" << endl;
    return;
  }
  address temp = Q->next;
  P = temp->info;
  Q->next = temp->next;
  deallocate(temp);
}

void updateAtPosition(List &L, int posisi, Product P) {
  if (isEmpty(L)) {
    cout << "List kosong!" << endl;
    return;
  }

  address temp = L.head;
  int idx = 1;

  while (temp != NULL && idx < posisi) {
    temp = temp->next;
    idx++;
  }

  if (temp == NULL) {
    cout << "Posisi " << posisi << " tidak ditemukan!" << endl;
    return;
  }

  Product oldData = temp->info;
  temp->info = P;

  cout << "\n=== Update Data Produk pada Posisi " << posisi << " ===" << endl;
  cout << "Data lama: " << oldData.Nama << " (SKU: " << oldData.SKU << ")" << endl;
  cout << "Data baru: " << P.Nama << " (SKU: " << P.SKU << ")" << endl;
  cout << "Data berhasil diupdate!" << endl;
}

void viewList(List L) {
  if (isEmpty(L)) {
    cout << "List kosong!" << endl;
    return;
  }
```

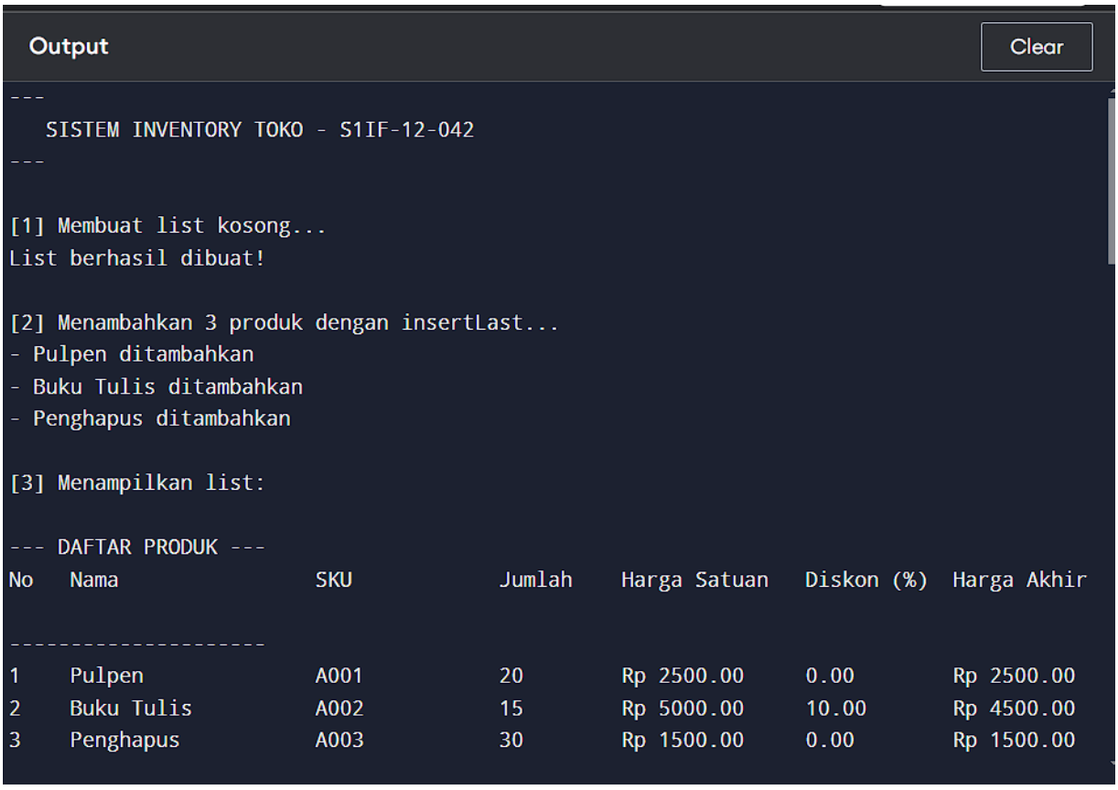
```
    cout << "\n--- DAFTAR PRODUK ---" << endl;
    cout << left << setw(5) << "No"
         << setw(20) << "Nama"
         << setw(15) << "SKU"
         << setw(10) << "Jumlah"
         << setw(15) << "Harga Satuan"
         << setw(12) << "Diskon (%)"
         << setw(15) << "Harga Akhir" << endl;
    cout << "----------------------" << endl;

    address temp = L.head;
    int idx = 1;

    while (temp != NULL) {
        float hargaAkhir = hitungHargaAkhir(temp->info);

        cout << left << setw(5) << idx
             << setw(20) << temp->info.Nama
             << setw(15) << temp->info.SKU
             << setw(10) << temp->info.Jumlah
             << "Rp " << setw(12) << fixed << setprecision(2) << temp->info.HargaSatuan
             << setw(12) << temp->info.DiskonPersen
             << "Rp " << fixed << setprecision(2) << hargaAkhir << endl;

        temp = temp->next;
        idx++;
    }
    cout << "----------------------" << endl;
}

void searchByFinalPriceRange(List L, float minPrice, float maxPrice) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    cout << "\n--- Pencarian Produk dengan Harga Akhir Rp " << fixed << setprecision(2)
         << minPrice << " - Rp " << maxPrice << " ---" << endl;

    address temp = L.head;
    int idx = 1;
    bool found = false;

    cout << left << setw(5) << "Pos"
         << setw(20) << "Nama"
         << setw(15) << "SKU"
         << setw(10) << "Jumlah"
         << setw(15) << "Harga Satuan"
         << setw(12) << "Diskon (%)"
         << setw(15) << "Harga Akhir" << endl;
    cout << "----------------------" << endl;

    while (temp != NULL) {
        float hargaAkhir = hitungHargaAkhir(temp->info);

        if (hargaAkhir >= minPrice && hargaAkhir <= maxPrice) {
            cout << left << setw(5) << idx
                 << setw(20) << temp->info.Nama
                 << setw(15) << temp->info.SKU
                 << setw(10) << temp->info.Jumlah
                 << "Rp " << setw(12) << fixed << setprecision(2) << temp->info.HargaSatuan
                 << setw(12) << temp->info.DiskonPersen
                 << "Rp " << fixed << setprecision(2) << hargaAkhir << endl;
```

```
            found = true;
        }

        temp = temp->next;
        idx++;
    }

    if (!found) {
        cout << "Tidak ada produk dalam rentang harga tersebut." << endl;
    }
    cout << "----------------------" << endl;
}

void MaxHargaAkhir(List L) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    address temp = L.head;
    float maxHarga = hitungHargaAkhir(temp->info);

    while (temp != NULL) {
        float hargaAkhir = hitungHargaAkhir(temp->info);
        if (hargaAkhir > maxHarga) {
            maxHarga = hargaAkhir;
        }
        temp = temp->next;
    }

    cout << "\n--- PRODUK HARGA AKHIR MAKSIMUM ---" << endl;
    cout << "Harga Akhir Maksimum: Rp " << fixed << setprecision(2) << maxHarga << endl;
    cout << "-------------------------------------" << endl;

    cout << left << setw(5) << "Pos"
         << setw(20) << "Nama"
         << setw(15) << "SKU"
         << setw(10) << "Jumlah"
         << setw(15) << "Harga Satuan"
         << setw(12) << "Diskon (%)"
         << setw(15) << "Harga Akhir" << endl;
    cout << "-------------------------------------" << endl;

    temp = L.head;
    int idx = 1;

    while (temp != NULL) {
        float hargaAkhir = hitungHargaAkhir(temp->info);

        if (hargaAkhir == maxHarga) {
            cout << left << setw(5) << idx
                 << setw(20) << temp->info.Nama
                 << setw(15) << temp->info.SKU
                 << setw(10) << temp->info.Jumlah
                 << "Rp " << setw(12) << fixed << setprecision(2) << temp->info.HargaSatuan
                 << setw(12) << temp->info.DiskonPersen
                 << "Rp " << fixed << setprecision(2) << hargaAkhir << endl;
        }

        temp = temp->next;
        idx++;
    }
    cout << "-------------------------------------" << endl;
```

```
}

// ==================== PROGRAM UTAMA ====================

int main() {
   List L;
   Product P;

   cout << "---" << endl;
   cout << "   SISTEM INVENTORY TOKO - S1IF-12-042" << endl;
   cout << "---" << endl;

   // 1. Buat list kosong
   cout << "\n[1] Membuat list kosong..." << endl;
   createList(L);
   cout << "List berhasil dibuat!" << endl;

   // 2. Insert Last 3 produk sesuai soal
   cout << "\n[2] Menambahkan 3 produk dengan insertLast..." << endl;

   Product p1;
   p1.Nama = "Pulpen";
   p1.SKU = "A001";
   p1.Jumlah = 20;
   p1.HargaSatuan = 2500;
   p1.DiskonPersen = 0;
   insertLast(L, p1);
   cout << "- Pulpen ditambahkan" << endl;

   Product p2;
   p2.Nama = "Buku Tulis";
   p2.SKU = "A002";
   p2.Jumlah = 15;
   p2.HargaSatuan = 5000;
   p2.DiskonPersen = 10;
   insertLast(L, p2);
   cout << "- Buku Tulis ditambahkan" << endl;

   Product p3;
   p3.Nama = "Penghapus";
   p3.SKU = "A003";
   p3.Jumlah = 30;
   p3.HargaSatuan = 1500;
   p3.DiskonPersen = 0;
   insertLast(L, p3);
   cout << "- Penghapus ditambahkan" << endl;

   // 3. Tampilkan list
   cout << "\n[3] Menampilkan list:" << endl;
   viewList(L);

   // 4. Delete First 1x
   cout << "\n[4] Melakukan deleteFirst 1x..." << endl;
   deleteFirst(L, P);
   cout << "Produk \"" << P.Nama << "\" (SKU: " << P.SKU << ") berhasil dihapus dari awal list." <<
endl;

   // 5. Tampilkan list kembali
   cout << "\n[5] Menampilkan list setelah deleteFirst:" << endl;
   viewList(L);

   // 6. Update data pada posisi ke-2
   cout << "\n[6] Update data pada posisi ke-2..." << endl;
```

```
    cout << "Data yang akan diinput:" << endl;
    cout << "Nama        : Stabilo" << endl;
    cout << "SKU         : A010" << endl;
    cout << "Jumlah      : 40" << endl;
    cout << "HargaSatuan  : 9000" << endl;
    cout << "DiskonPersen : 5" << endl;

    Product pUpdate;
    pUpdate.Nama = "Stabilo";
    pUpdate.SKU = "A010";
    pUpdate.Jumlah = 40;
    pUpdate.HargaSatuan = 9000;
    pUpdate.DiskonPersen = 5;

    updateAtPosition(L, 2, pUpdate);

    // 7. Tampilkan list setelah update
    cout << "\n[7] Menampilkan list setelah update:" << endl;
    viewList(L);

    // 8. Searching berdasarkan HargaAkhir dalam rentang min=2000, max=7000
    cout << "\n[8] Searching produk dengan HargaAkhir dalam rentang Rp 2000 - Rp 7000:" <<
endl;
    searchByFinalPriceRange(L, 2000, 7000);

    // BAGIAN B: Panggil MaxHargaAkhir
    cout << "\n[BAGIAN B] Menampilkan produk dengan HargaAkhir maksimum:" << endl;
    MaxHargaAkhir(L);

    cout << "\n========================================" << endl;
    cout << "    PROGRAM SELESAI" << endl;
    cout << "========================================" << endl;

    return 0;
}
```

Screenshot Output Program



```
[4] Melakukan deleteFirst 1x...
Produk "Pulpen" (SKU: A001) berhasil dihapus dari awal list.

[5] Menampilkan list setelah deleteFirst:

--- DAFTAR PRODUK ---
No   Nama               SKU            Jumlah    Harga Satuan   Diskon (%)  Harga Akhir

--------------------
1    Buku Tulis         A002           15        Rp 5000.00     10.00       Rp 4500.00
2    Penghapus          A003           30        Rp 1500.00     0.00        Rp 1500.00
--------------------

[6] Update data pada posisi ke-2...
Data yang akan diinput:
Nama         : Stabilo
SKU          : A010
Jumlah       : 40
HargaSatuan  : 9000
DiskonPersen : 5
```

```
=== Update Data Produk pada Posisi 2 ===
Data lama: Penghapus (SKU: A003)
Data baru: Stabilo (SKU: A010)
Data berhasil diupdate!

[7] Menampilkan list setelah update:

--- DAFTAR PRODUK ---
No   Nama               SKU            Jumlah    Harga Satuan   Diskon (%)  Harga Akhir
--------------------
1    Buku Tulis         A002           15        Rp 5000.00     10.00       Rp 4500.00
2    Stabilo            A010           40        Rp 9000.00     5.00        Rp 8550.00
--------------------

[8] Searching produk dengan HargaAkhir dalam rentang Rp 2000 - Rp 7000:

--- Pencarian Produk dengan Harga Akhir Rp 2000.00 - Rp 7000.00 ---
Pos  Nama               SKU            Jumlah    Harga Satuan   Diskon (%)  Harga Akhir
--------------------
1    Buku Tulis         A002           15        Rp 5000.00     10.00       Rp 4500.00
--------------------

[BAGIAN B] Menampilkan produk dengan HargaAkhir maksimum:

--- PRODUK HARGA AKHIR MAKSIMUM ---
Harga Akhir Maksimum: Rp 8550.00
-----------------------------------
Pos  Nama               SKU            Jumlah    Harga Satuan   Diskon (%)  Harga Akhir
-----------------------------------
2    Stabilo            A010           40        Rp 9000.00     5.00        Rp 8550.00
-----------------------------------
```

Soal Nomor 2 (Double Linked List)

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

// ===================== DEFINISI STRUCT =====================

struct Song {
   string Title;
   string Artist;
   int DurationSec;
   int PlayCount;
   float Rating;
};

struct Node;
typedef Node* address;

struct Node {
   Song info;
   address prev;
   address next;
};

struct List {
   address head;
   address tail;
};

// ===================== DEKLARASI FUNGSI/PROSEDUR =====================

float hitungPopularityScore(Song S);
bool isEmpty(List L);
void createList(List &L);
address allocate(Song S);
void deallocate(address P);
void insertFirst(List &L, Song S);
void insertLast(List &L, Song S);
void insertAfter(List &L, address Q, Song S);
void insertBefore(List &L, address Q, Song S);
void deleteFirst(List &L, Song &S);
void deleteLast(List &L, Song &S);
void deleteAfter(List &L, address Q, Song &S);
void deleteBefore(List &L, address Q, Song &S);
void updateAtPosition(List &L, int posisi, Song S);
void updateBefore(List &L, address Q, Song S);
void viewList(List L);
void searchByPopularityRange(List L, float minScore, float maxScore);
address getNodeAtPosition(List L, int posisi);

// ===================== IMPLEMENTASI FUNGSI/PROSEDUR
=====================

float hitungPopularityScore(Song S) {
   return 0.8 * S.PlayCount + 20.0 * S.Rating;
}

bool isEmpty(List L) {
   return L.head == NULL;
}

void createList(List &L) {
   L.head = NULL;
   L.tail = NULL;
}
```

```
address allocate(Song S) {
   address newNode = new Node;
   newNode->info = S;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

void deallocate(address P) {
   delete P;
}

void insertFirst(List &L, Song S) {
   address newNode = allocate(S);
   if (isEmpty(L)) {
      L.head = newNode;
      L.tail = newNode;
   } else {
      newNode->next = L.head;
      L.head->prev = newNode;
      L.head = newNode;
   }
}

void insertLast(List &L, Song S) {
   address newNode = allocate(S);
   if (isEmpty(L)) {
      L.head = newNode;
      L.tail = newNode;
   } else {
      newNode->prev = L.tail;
      L.tail->next = newNode;
      L.tail = newNode;
   }
}

void insertAfter(List &L, address Q, Song S) {
   if (Q == NULL) {
      cout << "Node Q tidak valid!" << endl;
      return;
   }

   address newNode = allocate(S);

   if (Q == L.tail) {
      newNode->prev = Q;
      Q->next = newNode;
      L.tail = newNode;
   } else {
      newNode->next = Q->next;
      newNode->prev = Q;
      Q->next->prev = newNode;
      Q->next = newNode;
   }
}

void insertBefore(List &L, address Q, Song S) {
   if (Q == NULL) {
      cout << "Node Q tidak valid!" << endl;
      return;
   }

   address newNode = allocate(S);

   if (Q == L.head) {
      newNode->next = Q;
```

```
        Q->prev = newNode;
        L.head = newNode;
    } else {
        newNode->prev = Q->prev;
        newNode->next = Q;
        Q->prev->next = newNode;
        Q->prev = newNode;
    }
}

void deleteFirst(List &L, Song &S) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    address temp = L.head;
    S = temp->info;

    if (L.head == L.tail) {
        L.head = NULL;
        L.tail = NULL;
    } else {
        L.head = L.head->next;
        L.head->prev = NULL;
    }

    deallocate(temp);
}

void deleteLast(List &L, Song &S) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    address temp = L.tail;
    S = temp->info;

    if (L.head == L.tail) {
        L.head = NULL;
        L.tail = NULL;
    } else {
        L.tail = L.tail->prev;
        L.tail->next = NULL;
    }

    deallocate(temp);
}

void deleteAfter(List &L, address Q, Song &S) {
    if (Q == NULL || Q->next == NULL) {
        cout << "Tidak ada node setelah Q!" << endl;
        return;
    }

    address temp = Q->next;
    S = temp->info;

    if (temp == L.tail) {
        L.tail = Q;
        Q->next = NULL;
    } else {
        Q->next = temp->next;
        temp->next->prev = Q;
    }
```

```
    deallocate(temp);
}

void deleteBefore(List &L, address Q, Song &S) {
    if (Q == NULL || Q->prev == NULL) {
        cout << "Tidak ada node sebelum Q!" << endl;
        return;
    }

    address temp = Q->prev;
    S = temp->info;

    if (temp == L.head) {
        L.head = Q;
        Q->prev = NULL;
    } else {
        temp->prev->next = Q;
        Q->prev = temp->prev;
    }

    deallocate(temp);
}

address getNodeAtPosition(List L, int posisi) {
    if (isEmpty(L)) {
        return NULL;
    }

    address temp = L.head;
    int idx = 1;

    while (temp != NULL && idx < posisi) {
        temp = temp->next;
        idx++;
    }

    return temp;
}

void updateAtPosition(List &L, int posisi, Song S) {
    address temp = getNodeAtPosition(L, posisi);

    if (temp == NULL) {
        cout << "Posisi " << posisi << " tidak ditemukan!" << endl;
        return;
    }

    Song oldData = temp->info;
    temp->info = S;

    cout << "\n--- Update Data Lagu pada Posisi " << posisi << " ---" << endl;
    cout << "Data lama: " << oldData.Title << " by " << oldData.Artist << endl;
    cout << "Data baru: " << S.Title << " by " << S.Artist << endl;
    cout << "Data berhasil diupdate!" << endl;
}

void updateBefore(List &L, address Q, Song S) {
    if (Q == NULL || Q->prev == NULL) {
        cout << "Tidak ada node sebelum Q!" << endl;
        return;
    }

    address temp = Q->prev;
    Song oldData = temp->info;
    temp->info = S;

    cout << "\n--- Update Data Lagu Sebelum Node ---" << endl;
```

```
    cout << "Data lama: " << oldData.Title << " by " << oldData.Artist << endl;
    cout << "Data baru: " << S.Title << " by " << S.Artist << endl;
    cout << "Data berhasil diupdate!" << endl;
}

void viewList(List L) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    cout << "\n--- DAFTAR LAGU PLAYLIST ---" << endl;
    cout << left << setw(5) << "Pos"
         << setw(20) << "Title"
         << setw(15) << "Artist"
         << setw(10) << "Duration"
         << setw(10) << "PlayCount"
         << setw(10) << "Rating"
         << setw(15) << "PopScore" << endl;
    cout << "----------------------" << endl;

    address temp = L.head;
    int idx = 1;

    while (temp != NULL) {
        float popScore = hitungPopularityScore(temp->info);

        cout << left << setw(5) << idx
             << setw(20) << temp->info.Title
             << setw(15) << temp->info.Artist
             << setw(10) << temp->info.DurationSec
             << setw(10) << temp->info.PlayCount
             << setw(10) << fixed << setprecision(1) << temp->info.Rating
             << setw(15) << fixed << setprecision(2) << popScore << endl;

        temp = temp->next;
        idx++;
    }
    cout << "----------------------" << endl;
}

void searchByPopularityRange(List L, float minScore, float maxScore) {
    if (isEmpty(L)) {
        cout << "List kosong!" << endl;
        return;
    }

    cout << "\n--- Pencarian Lagu dengan PopularityScore " << fixed << setprecision(2)
         << minScore << " - " << maxScore << " ---" << endl;

    address temp = L.head;
    int idx = 1;
    bool found = false;

    cout << left << setw(5) << "Pos"
         << setw(20) << "Title"
         << setw(15) << "Artist"
         << setw(10) << "Duration"
         << setw(10) << "PlayCount"
         << setw(10) << "Rating"
         << setw(15) << "PopScore" << endl;
    cout << "----------------------" << endl;

    while (temp != NULL) {
        float popScore = hitungPopularityScore(temp->info);

        if (popScore >= minScore && popScore <= maxScore) {
```

```
            cout << left << setw(5) << idx
                 << setw(20) << temp->info.Title
                 << setw(15) << temp->info.Artist
                 << setw(10) << temp->info.DurationSec
                 << setw(10) << temp->info.PlayCount
                 << setw(10) << fixed << setprecision(1) << temp->info.Rating
                 << setw(15) << fixed << setprecision(2) << popScore << endl;
            found = true;
        }

        temp = temp->next;
        idx++;
    }

    if (!found) {
        cout << "Tidak ada lagu dalam rentang PopularityScore tersebut." << endl;
    }
    cout << "----------------------" << endl;
}

// ====================== PROGRAM UTAMA ======================

int main() {
    List L;
    Song S;

    cout << "---" << endl;
    cout << "   SISTEM PLAYLIST LAGU - DOUBLE LINKED LIST" << endl;
    cout << "---" << endl;

    // 1. Buat list kosong
    cout << "\n[1] Membuat list kosong..." << endl;
    createList(L);
    cout << "List berhasil dibuat!" << endl;

    // 2. Insert Last 3 lagu
    cout << "\n[2] Menambahkan 3 lagu dengan insertLast..." << endl;

    Song s1;
    s1.Title = "Senja di Kota";
    s1.Artist = "Nona Band";
    s1.DurationSec = 210;
    s1.PlayCount = 150;
    s1.Rating = 4.2;
    insertLast(L, s1);
    cout << "- Senja di Kota ditambahkan" << endl;

    Song s2;
    s2.Title = "Langkahmu";
    s2.Artist = "Delta";
    s2.DurationSec = 185;
    s2.PlayCount = 320;
    s2.Rating = 4.8;
    insertLast(L, s2);
    cout << "- Langkahmu ditambahkan" << endl;

    Song s3;
    s3.Title = "Hujan Minggu";
    s3.Artist = "Arka";
    s3.DurationSec = 240;
    s3.PlayCount = 90;
    s3.Rating = 3.9;
    insertLast(L, s3);
    cout << "- Hujan Minggu ditambahkan" << endl;

    // 3. Tampilkan list
    cout << "\n[3] Menampilkan list:" << endl;
```

```
viewList(L);

// 4. Delete Last 1x
cout << "\n[4] Melakukan deleteLast 1x..." << endl;
deleteLast(L, S);
cout << "Lagu \"" << S.Title << "\" by " << S.Artist << " berhasil dihapus." << endl;
viewList(L);

// 5. Update pada posisi ke-2
cout << "\n[5] Update node pada posisi ke-2..." << endl;
cout << "Data yang akan diinput:" << endl;
cout << "Title        : Pelita" << endl;
cout << "Artist       : Luna" << endl;
cout << "DurationSec  : 200" << endl;
cout << "PlayCount    : 260" << endl;
cout << "Rating       : 4.5" << endl;

Song sUpdate;
sUpdate.Title = "Pelita";
sUpdate.Artist = "Luna";
sUpdate.DurationSec = 200;
sUpdate.PlayCount = 260;
sUpdate.Rating = 4.5;

updateAtPosition(L, 2, sUpdate);

// 6. Tampilkan list setelah update
cout << "\n[6] Menampilkan list setelah update:" << endl;
viewList(L);

// 7. Operasi BEFORE
cout << "\n[7] OPERASI BEFORE" << endl;

// 7a. insertBefore pada node posisi 2
cout << "\n[7a] insertBefore pada node posisi 2..." << endl;
address nodePos2 = getNodeAtPosition(L, 2);

Song sInsert;
sInsert.Title = "Senandung";
sInsert.Artist = "Mira";
sInsert.DurationSec = 175;
sInsert.PlayCount = 120;
sInsert.Rating = 4.0;

insertBefore(L, nodePos2, sInsert);
cout << "Lagu \"Senandung\" berhasil ditambahkan sebelum posisi 2." << endl;
viewList(L);

// 7b. updateBefore pada node posisi 2
cout << "\n[7b] updateBefore pada node posisi 2..." << endl;
nodePos2 = getNodeAtPosition(L, 2);

Song sUpdateBefore;
sUpdateBefore.Title = "Cahaya Baru";
sUpdateBefore.Artist = "Rama";
sUpdateBefore.DurationSec = 195;
sUpdateBefore.PlayCount = 180;
sUpdateBefore.Rating = 4.3;

updateBefore(L, nodePos2, sUpdateBefore);
viewList(L);

// 7c. deleteBefore pada node posisi 3
cout << "\n[7c] deleteBefore pada node posisi 3..." << endl;
address nodePos3 = getNodeAtPosition(L, 3);

deleteBefore(L, nodePos3, S);
```

```
    cout << "Lagu \"" << S.Title << "\" by " << S.Artist << " berhasil dihapus sebelum posisi 3."
<< endl;
    viewList(L);

    // 8. Searching berdasarkan PopularityScore
    cout << "\n[8] Searching berdasarkan PopularityScore (150.0 - 300.0):" << endl;
    searchByPopularityRange(L, 150.0, 300.0);

    cout << "\n---" << endl;
    cout << "   PROGRAM SELESAI" << endl;
    cout << "---" << endl;

    return 0;
}
```

Screenshot Program

Output

```
---
   SISTEM PLAYLIST LAGU - DOUBLE LINKED LIST
---

[1] Membuat list kosong...
List berhasil dibuat!

[2] Menambahkan 3 lagu dengan insertLast...
- Senja di Kota ditambahkan
- Langkahmu ditambahkan
- Hujan Minggu ditambahkan

[3] Menampilkan list:

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
---------------------
1    Senja di Kota       Nona Band      210       150       4.2       204.00
2    Langkahmu           Delta          185       320       4.8       352.00
3    Hujan Minggu        Arka           240       90        3.9       150.00
---------------------

[4] Melakukan deleteLast 1x...
Lagu "Hujan Minggu" by Arka berhasil dihapus.

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
---------------------
1    Senja di Kota       Nona Band      210       150       4.2       204.00
2    Langkahmu           Delta          185       320       4.8       352.00
---------------------
```

Output

```
--------------------

[5] Update node pada posisi ke-2...
Data yang akan diinput:
Title        : Pelita
Artist       : Luna
DurationSec  : 200
PlayCount    : 260
Rating       : 4.5

--- Update Data Lagu pada Posisi 2 ---
Data lama: Langkahmu by Delta
Data baru: Pelita by Luna
Data berhasil diupdate!

[6] Menampilkan list setelah update:

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
--------------------
1    Senja di Kota       Nona Band      210       150       4.2       204.00
2    Pelita              Luna           200       260       4.5       298.00
--------------------

[7] OPERASI BEFORE

[7a] insertBefore pada node posisi 2...
Lagu "Senandung" berhasil ditambahkan sebelum posisi 2.

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
--------------------
1    Senja di Kota       Nona Band      210       150       4.2       204.00
2    Senandung           Mira           175       120       4.0       176.00
3    Pelita              Luna           200       260       4.5       298.00
--------------------
```

Output

```
[7b] updateBefore pada node posisi 2...

--- Update Data Lagu Sebelum Node ---
Data lama: Senja di Kota by Nona Band
Data baru: Cahaya Baru by Rama
Data berhasil diupdate!

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
---------------------
1    Cahaya Baru        Rama           195       180       4.3       230.00
2    Senandung          Mira           175       120       4.0       176.00
3    Pelita             Luna           200       260       4.5       298.00
---------------------

[7c] deleteBefore pada node posisi 3...
Lagu "Senandung" by Mira berhasil dihapus sebelum posisi 3.

--- DAFTAR LAGU PLAYLIST ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
---------------------
1    Cahaya Baru        Rama           195       180       4.3       230.00
2    Pelita             Luna           200       260       4.5       298.00
---------------------

[8] Searching berdasarkan PopularityScore (150.0 - 300.0):

--- Pencarian Lagu dengan PopularityScore 150.00 - 300.00 ---
Pos  Title              Artist         Duration  PlayCount Rating    PopScore
---------------------
1    Cahaya Baru        Rama           195       180       4.3       230.00
2    Pelita             Luna           200       260       4.5       298.00
---------------------

---
   PROGRAM SELESAI
---
```

Soal Nomor 3 (Stack)

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

struct Mahasiswa {
   string Nama;
   string NIM;
   float NilaiTugas;
   float NilaiUTS;
   float NilaiUAS;
};

const int MAX = 6;

struct StackMahasiswa {
   Mahasiswa dataMahasiswa[MAX];
   int Top;
};

bool isEmpty(StackMahasiswa &StackMHS) {
   return StackMHS.Top == -1;
}

bool isFull(StackMahasiswa &StackMHS) {
   return StackMHS.Top == MAX - 1;
}

void createStack(StackMahasiswa &StackMHS) {
   StackMHS.Top = -1;
}

void Push(StackMahasiswa &StackMHS, Mahasiswa dataBaru) {
   if (isFull(StackMHS)) {
      cout << "Stack penuh!\n";
      return;
   }
   StackMHS.Top++;
   StackMHS.dataMahasiswa[StackMHS.Top] = dataBaru;
}

void Pop(StackMahasiswa &StackMHS) {
   if (isEmpty(StackMHS)) {
      cout << "Stack kosong!\n";
      return;
   }
   StackMHS.Top--;
}

void Update(StackMahasiswa &StackMHS, int posisi, Mahasiswa dataBaru) {
   if (posisi < 1 || posisi > StackMHS.Top + 1) {
      cout << "Posisi tidak valid!\n";
      return;
   }
   StackMHS.dataMahasiswa[posisi - 1] = dataBaru;
}

float hitungNilaiAkhir(Mahasiswa M) {
   return 0.2 * M.NilaiTugas + 0.4 * M.NilaiUTS + 0.4 * M.NilaiUAS;
}

void View(StackMahasiswa &StackMHS) {
   if (isEmpty(StackMHS)) {
```

```
        cout << "Stack kosong!\n";
        return;
    }
    for (int i = StackMHS.Top; i >= 0; i--) {
        Mahasiswa &m = StackMHS.dataMahasiswa[i];
        cout << "Posisi " << i+1 << ":\n";
        cout << "Nama: " << m.Nama << "\n";
        cout << "NIM: " << m.NIM << "\n";
        cout << "Nilai Tugas: " << m.NilaiTugas << "\n";
        cout << "Nilai UTS: " << m.NilaiUTS << "\n";
        cout << "Nilai UAS: " << m.NilaiUAS << "\n";
        cout << "Nilai Akhir: " << hitungNilaiAkhir(m) << "\n\n";
    }
}

void SearchNilaiAkhir(StackMahasiswa &StackMHS, float Min, float Max) {
    bool found = false;

    for (int i = 0; i <= StackMHS.Top; i++) {
        float nilai = hitungNilaiAkhir(StackMHS.dataMahasiswa[i]);
        if (nilai >= Min && nilai <= Max) {
            found = true;
            cout << "Ditemukan pada posisi " << i + 1 << ":\n";
            cout << "Nama: " << StackMHS.dataMahasiswa[i].Nama << "\n";
            cout << "NIM: " << StackMHS.dataMahasiswa[i].NIM << "\n";
            cout << "Nilai Akhir: " << nilai << "\n\n";
        }
    }

    if (!found) {
        cout << "Tidak ada mahasiswa dengan nilai dalam rentang tersebut.\n";
    }
}

void MaxNilaiAkhir(StackMahasiswa &StackMHS) {
    if (isEmpty(StackMHS)) {
        cout << "Stack kosong!\n";
        return;
    }

    int idxMax = 0;
    float maxNilai = hitungNilaiAkhir(StackMHS.dataMahasiswa[0]);

    for (int i = 1; i <= StackMHS.Top; i++) {
        float nilai = hitungNilaiAkhir(StackMHS.dataMahasiswa[i]);
        if (nilai > maxNilai) {
            maxNilai = nilai;
            idxMax = i;
        }
    }

    cout << "Mahasiswa dengan nilai akhir tertinggi:\n";
    cout << "Posisi: " << idxMax + 1 << "\n";
    cout << "Nama: " << StackMHS.dataMahasiswa[idxMax].Nama << "\n";
    cout << "NIM: " << StackMHS.dataMahasiswa[idxMax].NIM << "\n";
    cout << "Nilai Akhir: " << maxNilai << "\n\n";
}

int main() {

    StackMahasiswa S;
    createStack(S);
```

```
    Mahasiswa M1 = {"Venti", "11111", 75.7, 82.1, 65.5};
    Mahasiswa M2 = {"Xiao", "22222", 87.4, 88.9, 81.9};
    Mahasiswa M3 = {"Kazuha", "33333", 92.3, 88.8, 82.4};
    Mahasiswa M4 = {"Wanderer7", "44444", 95.5, 85.5, 90.5};
    Mahasiswa M5 = {"Lynette", "55555", 77.7, 65.4, 79.9};
    Mahasiswa M6 = {"Chasca", "66666", 99.9, 93.6, 87.3};

    Push(S, M1);
    Push(S, M2);
    Push(S, M3);
    Push(S, M4);
    Push(S, M5);
    Push(S, M6);

    cout << "=== DATA STACK AWAL ===\n";
    View(S);

    Pop(S);

    cout << "=== DATA SETELAH POP 1X ===\n";
    View(S);

    Mahasiswa Up = {"Heizou", "77777", 99.9, 88.8, 77.7};
    Update(S, 3, Up);

    cout << "=== DATA SETELAH UPDATE POSISI 3 ===\n";
    View(S);

    cout << "=== SEARCH NILAI AKHIR (85.5 - 95.5) ===\n";
    SearchNilaiAkhir(S, 85.5, 95.5);

    cout << "=== MAHASISWA NILAI AKHIR TERTINGGI ===\n";
    MaxNilaiAkhir(S);

    return 0;
}
```

Screenshot Output Program

Output

```
=== DATA STACK AWAL ===
Posisi 6:
Nama: Chasca
NIM: 66666
Nilai Tugas: 99.9
Nilai UTS: 93.6
Nilai UAS: 87.3
Nilai Akhir: 92.34

Posisi 5:
Nama: Lynette
NIM: 55555
Nilai Tugas: 77.7
Nilai UTS: 65.4
Nilai UAS: 79.9
Nilai Akhir: 73.66

Posisi 4:
Nama: Wanderer7
NIM: 44444
Nilai Tugas: 95.5
Nilai UTS: 85.5
Nilai UAS: 90.5
Nilai Akhir: 89.5

Posisi 3:
Nama: Kazuha
NIM: 33333
Nilai Tugas: 92.3
Nilai UTS: 88.8
Nilai UAS: 82.4
Nilai Akhir: 86.94

Posisi 2:
Nama: Xiao
NIM: 22222
Nilai Tugas: 87.4
Nilai UTS: 88.9
Nilai UAS: 81.9
Nilai Akhir: 85.8

Posisi 1:
Nama: Venti
NIM: 11111
Nilai Tugas: 75.7
Nilai UTS: 82.1
Nilai UAS: 65.5
Nilai Akhir: 74.18
```

Output

```
=== DATA SETELAH POP 1X ===
Posisi 5:
Nama: Lynette
NIM: 55555
Nilai Tugas: 77.7
Nilai UTS: 65.4
Nilai UAS: 79.9
Nilai Akhir: 73.66

Posisi 4:
Nama: Wanderer7
NIM: 44444
Nilai Tugas: 95.5
Nilai UTS: 85.5
Nilai UAS: 90.5
Nilai Akhir: 89.5

Posisi 3:
Nama: Kazuha
NIM: 33333
Nilai Tugas: 92.3
Nilai UTS: 88.8
Nilai UAS: 82.4
Nilai Akhir: 86.94

Posisi 2:
Nama: Xiao
NIM: 22222
Nilai Tugas: 87.4
Nilai UTS: 88.9
Nilai UAS: 81.9
Nilai Akhir: 85.8

Posisi 1:
Nama: Venti
NIM: 11111
Nilai Tugas: 75.7
```

Output

```
=== DATA SETELAH UPDATE POSISI 3 ===
Posisi 5:
Nama: Lynette
NIM: 55555
Nilai Tugas: 77.7
Nilai UTS: 65.4
Nilai UAS: 79.9
Nilai Akhir: 73.66

Posisi 4:
Nama: Wanderer7
NIM: 44444
Nilai Tugas: 95.5
Nilai UTS: 85.5
Nilai UAS: 90.5
Nilai Akhir: 89.5

Posisi 3:
Nama: Heizou
NIM: 77777
Nilai Tugas: 99.9
Nilai UTS: 88.8
Nilai UAS: 77.7
Nilai Akhir: 86.58

Posisi 2:
Nama: Xiao
NIM: 22222
Nilai Tugas: 87.4
Nilai UTS: 88.9
Nilai UAS: 81.9
Nilai Akhir: 85.8

Posisi 1:
Nama: Venti
NIM: 11111
Nilai Tugas: 75.7
Nilai UTS: 82.1
```

```
=== SEARCH NILAI AKHIR (85.5 - 95.5) ===
Ditemukan pada posisi 2:
Nama: Xiao
NIM: 22222
Nilai Akhir: 85.8

Ditemukan pada posisi 3:
Nama: Heizou
NIM: 77777
Nilai Akhir: 86.58

Ditemukan pada posisi 4:
Nama: Wanderer7
NIM: 44444
Nilai Akhir: 89.5

=== MAHASISWA NILAI AKHIR TERTINGGI ===
Mahasiswa dengan nilai akhir tertinggi:
Posisi: 4
Nama: Wanderer7
NIM: 44444
Nilai Akhir: 89.5
```

Soal Nomor 4 (Queue)

Sourcode

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

// ============================
//       STRUCT & ADT
// ============================

struct Paket {
   string KodeResi;
   string NamaPengirim;
   int BeratBarang;
   string Tujuan;
};

const int MAX = 5;

struct QueueEkspedisi {
   Paket dataPaket[MAX];
   int Head;
   int Tail;
};

bool isEmpty(QueueEkspedisi &Q) {
   return (Q.Head == -1 && Q.Tail == -1);
}

bool isFull(QueueEkspedisi &Q) {
   return (Q.Tail == MAX - 1);
}

void createQueue(QueueEkspedisi &Q) {
   Q.Head = -1;
   Q.Tail = -1;
}

void enQueue(QueueEkspedisi &Q, Paket dataBaru) {
   if (isFull(Q)) {
      cout << "Queue penuh!" << endl;
      return;
   }

   if (isEmpty(Q)) {
      Q.Head = 0;
      Q.Tail = 0;
      Q.dataPaket[0] = dataBaru;
   } else {
      Q.Tail++;
      Q.dataPaket[Q.Tail] = dataBaru;
   }
}

void deQueue(QueueEkspedisi &Q) {
   if (isEmpty(Q)) {
      cout << "Queue kosong!" << endl;
      return;
   }

   if (Q.Head == Q.Tail) {
```

```
        createQueue(Q);
    } else {
        Q.Head++;
    }
}

void viewQueue(QueueEkspedisi &Q) {
    if (isEmpty(Q)) {
        cout << "Queue kosong!" << endl;
        return;
    }

    for (int i = Q.Head; i <= Q.Tail; i++) {
        cout << "Posisi " << i + 1 << endl;
        cout << "Kode Resi      : " << Q.dataPaket[i].KodeResi << endl;
        cout << "Nama Pengirim  : " << Q.dataPaket[i].NamaPengirim << endl;
        cout << "Berat Barang   : " << Q.dataPaket[i].BeratBarang << " kg" << endl;
        cout << "Tujuan         : " << Q.dataPaket[i].Tujuan << endl << endl;
    }
}

int TotalBiayaPengiriman(QueueEkspedisi &Q) {
    if (isEmpty(Q)) return 0;

    int totalBerat = 0;
    for (int i = Q.Head; i <= Q.Tail; i++) {
        totalBerat += Q.dataPaket[i].BeratBarang;
    }
    return totalBerat * 8250;
}

// ============================
//          MAIN
// ============================

int main() {
    QueueEkspedisi Q;
    createQueue(Q);

    int pilihan;

    do {
        cout << "\n--- Komaniya Ekspress ---\n";
        cout << "1. Input Data Paket\n";
        cout << "2. Hapus Data Paket (deQueue)\n";
        cout << "3. Tampilkan Queue Paket\n";
        cout << "4. Hitung Total Biaya Pengiriman\n";
        cout << "0. Keluar\n";
        cout << "Pilihan anda : ";
        cin >> pilihan;

        if (pilihan == 1) {
            Paket P;
            cout << "Kode Resi      : "; cin >> P.KodeResi;
            cout << "Nama Pengirim  : "; cin >> P.NamaPengirim;
            cout << "Berat Barang   : "; cin >> P.BeratBarang;
            cout << "Tujuan         : "; cin >> P.Tujuan;
            enQueue(Q, P);
        }

        else if (pilihan == 2) {
            deQueue(Q);
        }
```

```
    else if (pilihan == 3) {
        viewQueue(Q);
    }

    else if (pilihan == 4) {
        int total = TotalBiayaPengiriman(Q);
        cout << "Total Biaya Pengiriman : Rp " << total << endl;
    }

  } while (pilihan != 0);

  return 0;
}
```

Screenshot Output Program

```
Output

--- Komaniya Ekspress ---
1. Input Data Paket
2. Hapus Data Paket (deQueue)
3. Tampilkan Queue Paket
4. Hitung Total Biaya Pengiriman
0. Keluar
Pilihan anda : 1
Kode Resi      : 123456
Nama Pengirim  : Hutao
Berat Barang   : 14
Tujuan         : Sumeru

--- Komaniya Ekspress ---
1. Input Data Paket
2. Hapus Data Paket (deQueue)
3. Tampilkan Queue Paket
4. Hitung Total Biaya Pengiriman
0. Keluar
Pilihan anda : 4
Total Biaya Pengiriman : Rp 115500
```

```
Pilihan anda : 1
Kode Resi       : 345678
Nama Pengirim   : Bennet
Berat Barang    : 7
Tujuan          : Natlan

--- Komaniya Ekspress ---
1. Input Data Paket
2. Hapus Data Paket (deQueue)
3. Tampilkan Queue Paket
4. Hitung Total Biaya Pengiriman
0. Keluar
Pilihan anda : 3
Posisi 1
Kode Resi       : 345678
Nama Pengirim   : Bennet
Berat Barang    : 7 kg
Tujuan          : Natlan
```