Rifqi Akhmad Maulana

5027211035

Laporan Praktikum Final Strukdat

1. Bantu Urutin Dong

Dalam soal ini diperintahkan untuk printing angka urut sesuai dengan indexnya, dimulai dengan index ganjil terlebih dahulu dan diikuti dengan genap.

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

int main()
{
    int n;
    int j = 0;
    int k = 0;
    cin >> n;

    int arr[n];
    int ganjil[n / 2];
    int genap[n / 2];

    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }

    for (int i = 0; i < n; i++)
    {
        if (i % 2 == 0)
        {
            ganjil[j] = arr[i];
            j++;
        }
        else
        {
            genap[k] = arr[i];
            k++;
        }
    }

    for (int i = 0; i < (j); i++)
    {
        cout << ganjil[i] << " ";
    }
    for (int i = 0; i < (k); i++)
    {
        cout << genap[i] << " ";
    }

    return 0;
}
```

Dengan menggunakan beberapa for loops, semua input diletakkan sementara dalam array kosong lalu ke 2 array lagi untuk membedakan angka genap dan ganjil sehingga memudahkan saat proses printing

2. Daily Temperatures

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

int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    for (int i = 0; i < n; i++)
    {
        int hari = 0;
        bool nemu = false;
        int temparr[n] = {};
        for (int j = (i+1); j < n; j++)
        {
            if (arr[j] < arr[i])
            {
                temparr[hari] = arr[j];
                hari++;
                continue;
            }
            else if (arr[j] > arr[i])
            {
                temparr[hari] = arr[j];
                hari++;
                nemu = true;
                break;
            }
        }
        if (nemu == true)
        {
            cout << hari << " ";
            for (int k = 0; temparr[k] != 0; k++)
            {
                cout << temparr[k] << " ";
            }
            cout << endl;
        }
        else if (nemu == false)
        {
            cout << "letsgo!!" << endl;
        }
    }
}
```

Menggunakan aplikasi for loop sederhana, pada for loop kedua untuk printing hari, terdapat reinisialisasi array kosong karena array tsb berguna untuk tempat menampung temperatur hari sebelum di print. Dengan nested for loop, loop mencari angka yang lebih besar dari angka array sebelumnya hingga ketemu,

3. Cek Komposisi Lagi Skuy

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

vector<int> arr;
vector<double> rata;
int ratarata;

void process(int komp)
{
    arr.push_back(komp);
}

void gass()
{
    double temp = 0;
    ratarata = 0;
    vector<int>::iterator it;
    for (it = arr.begin(); it != arr.end(); ++it)
    {
        ratarata++;
        temp += *it;
    }
    rata.push_back(temp/arr.size());

    cout << setprecision(2) << fixed << temp / (arr.size());
    cout << endl;
}

int main()
{
    int a;
    double b;
    cin >> a;
    cin >> b;

    double res = 0;
    double final = 0;
    for (int i = 0; i < a; i++)
    {
        string tempstr;
        int tempint;

        cin >> tempstr;

        if (tempstr == "GASS")
        {
            gass();
        }
        else
        {
            cin >> tempint;
            process(tempint);
        }
    }

    vector<double>::iterator it;
    for (it = rata.begin(); it != rata.end(); ++it)
    {
        res += *it;
    }

    final = res/(rata.size());
    cout << final << " ";

    if ((b * 0.5) < final && (b * 1.5) >= final)
    {
        cout << "AMAN";
    }
    else
    {
        cout << "LOH";
    }
    return 0;
}
```

Menggunakan C++ STL library. Setelah input jumlah testcase dan angka patokan gizi B, inputan diproses di fungsi, tergantung dengan apa yang diinput. Jika yang diinput adalah string “GASS” maka akan melakukan fungsi gass() yang akan melakukan printing rata-rata angka gizi sekarang. Selain dari itu akan masuk ke fungsi process. Angka gizi yang di inputkan setelah nama makanan/bahan akan dimasukkan ke array global yang kemudian nanti akan dioperasikan sebagaimana mestinya seperti yang tertera di kode di atas

4. Urutin oi

```cpp
#include <bits/stdc++.h>
#include <vector>
#include <iterator>
#include <algorithm>

using namespace std;

void remove(std::vector<int> &v)
{
    auto end = v.end();
    for (auto it = v.begin(); it != end; ++it) {
        end = std::remove(it + 1, end, *it);
    }

    v.erase(end, v.end());
}

int main() {
    vector<int> arr;

    while(true)
    {
        int val;
        cin >> val;
        if (val == 0)
        {
            break;
        }
        else
        {
            arr.push_back(val);
        }
    }

    remove(arr);
    vector<int>::iterator it;
    sort(arr.begin(), arr.end());
    for (it = arr.begin(); it != arr.end(); ++it)
    {
        cout << *it << endl;
    }
}
```

Menggunakan C++ STL Library, disini didapatkan solusi dengan menggunakan fungsi bawaan dari STL. Dapat dibuat input menggunakan while loop agar terus menerima dan berhenti sampai menemui angka 0, jika belum menemui angka 0 maka push_back ke vector yang sudah terbuat. Setelahnya, kita ingin membersihkan vector dari nilai duplikat, maka dengan itu harus dijalankan fungsi terlebih dahulu, lalu di sorting dan di print.

5. Doorprize toko ARA

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  int n;
  cin >> n;

  int total_prize = 0;

  vector<int> receipts;

  for (int i = 0; i < n; i++) {
    int num_receipts;
    cin >> num_receipts;

    for (int j = 0; j < num_receipts; j++) {
      int transaksi;
      cin >> transaksi;

      receipts.push_back(transaksi);
    }

    sort(receipts.begin(), receipts.end());

    if (receipts.size() >= 2) {
      total_prize += receipts[receipts.size() - 1] - receipts[0];
      receipts.pop_back();
      receipts.erase(receipts.begin());
    }
  }

  // Cetak total hadiah
  cout << total_prize << endl;

  return 0;
}
```

Menggunakan nested for loop untuk menentukan banyaknya entry pada hari tertentu dan memasukkan entry yang bersangkutan. Dari input entry lalu dimasukkan ke vector dan di sorting agar memudahkan proses penyelisihan. Setelahnya, entry terbesar dan terkecil di pop dan kembali ke awal loop, memulai lagi input entry dengan entry sisa yang sudah ada di iterasi sebelumnya.

6. Cinta Segitiga lagi nich

```cpp
#include <iostream>

using namespace std;

bool cts(int a, int b, int c, int n, int computers[]) {
  return computers[a - 1] == b && computers[b - 1] == c && computers[c - 1] == a;
}

int main() {
  int n;
  cin >> n;

  int computers[n];
  for (int i = 0; i < n; i++) {
    cin >> computers[i];
  }

  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      for (int k = 1; k <= n; k++) {
        if (cts(i, j, k, n, computers)) {
          cout << "YES";
          return 0;
        }
      }
    }
  }
  cout << "NO";
  return 0;
}
```

Dengan menggunakan 2 nested for loop untuk memeriksa nilai dari tiap index di array, dapat diperiksa 3 nilai secara bersamaan. Dalam fungsi, diperiksa apakah nilai dari index A array sama dengan nilai index B, dan diperiksa apakah nilai dari index b array sama dengan nilai index C. Jika iya, maka keluarkan output “YES”, sebaliknya keluarkan output “NO”

7. Seminar 1

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

#define n 100002

int R, C;
int val[n], pref[n], sum;

int getArrayIndex(int i, int j) {
    if(i == 0 || j == 0) return 0;
    return (i - 1) * C  + j;
}

pair<int, int> getCoordinateIndex(int pos) {
    return {(pos - 1) / C + 1 , (pos - 1) % C + 1};
}

pair<int, int> computeArea(int l1, int r1, int l2, int r2) {
    if(l1 <= 0 || r1 <= 0) return {0, 0};
    int area = (r2 - r1 + 1) * (l2 - l1 + 1);

    return {area, pref[getArrayIndex(l2, r2)] - pref[getArrayIndex(l2, r1 - 1)] - pref[getArrayIndex(l1 - 1, r2)]
+ pref[getArrayIndex(l1 - 1, r1 - 1)]};
}

int main() {
    cin >> R >> C;

    for(int i = 1; i <= R * C; i++) {
        char ch;
        cin >> ch;
        if(ch == '1') val[i] = 1;
        sum += val[i];
    }

    for(int i = 1; i <= R; i++) {
        for(int j = 1; j <= C; j++) {
            pref[getArrayIndex(i, j)] =  val[getArrayIndex(i, j)] + pref[getArrayIndex(i - 1, j)] +
pref[getArrayIndex(i, j - 1)] - pref[getArrayIndex(i - 1, j - 1)];
        }
    }

    vector<int> listDivider;

    for(int i = 1; i <= sum; i++) {
        if (sum % i == 0) {
            listDivider.push_back(i);
        }
    }

    int mini = 1e9;

    for(int i = 1; i <= R; i++) {
        for(int j = 1; j <= C; j++) {
            for(int k = 0; k < listDivider.size(); k++) {
                int divider = listDivider[k];
                int anotherDivider = sum / divider;

                int l1 = i - divider + 1, r1 = j - anotherDivider + 1;

                pair<int,int> tmp = computeArea(l1, r1, i , j);

                if(tmp.first == sum) {
                    mini = min(mini, sum - tmp.second);
                }
            }
        }
    }

    if (mini == 1e9) mini = -1;

    cout << mini << "\n";
}
```

Pertama membaca dimensi matriks (R dan C) dan kemudian membaca nilai untuk setiap sel dalam matriks. Jika sebuah sel memiliki nilai '1', nilai yang sesuai dalam array val diatur ke 1, dan variabel

penjumlahan bertambah. Kemudian menggunakan array jumlah awalan dua dimensi, pref, untuk menyimpan jumlah sub-matriks dalam matriks asli.

Selanjutnya, kode menghasilkan daftar pembagi untuk jumlah total 1 dalam matriks (disimpan dalam variabel sum) dan menyimpannya dalam array listDivider. Kemudian memasuki nested loop yang berulang pada setiap sel dalam matriks. Untuk setiap sel, ia mengulang setiap pembagi dalam larik listDivider, dan menggunakan sel saat ini sebagai sudut kanan atas persegi panjang. Kemudian menghitung sudut kiri bawah persegi panjang dengan mengurangkan pembagi saat ini dan pembagi yang sesuai (jumlah yang dibagi dengan pembagi saat ini) dari indeks baris dan kolom dari sudut kanan atas, masing-masing.

Kode kemudian menghitung jumlah 1 dalam persegi panjang menggunakan larik pref, dan memeriksa apakah itu sama dengan jumlah total 1 dalam matriks. Jika ya, kode memperbarui jumlah minimum 1 yang perlu dihapus untuk mencapai partisi menjadi dua sub-matriks dengan jumlah 1 yang sama. Setelah nested loop selesai, kode memeriksa apakah jumlah minimum 1 yang perlu dihapus telah diperbarui. Jika belum, jumlah minimum dari 1 diatur ke -1, yang menunjukkan bahwa tidak mungkin ada partisi. Kode kemudian menampilkan jumlah minimum 1 yang perlu dihapus.

8. Program Canggih

(Screenshot dan sc terlampir)

Pada testcase ini, pertama-tama ada perintah insert 3, sehingga AVL Tree memiliki satu node dengan nilai 3. Kemudian ada perintah insert 1, sehingga AVL Tree memiliki dua node dengan nilai 3 dan 1. Node dengan nilai 1 ditambahkan sebagai anak kiri dari node dengan nilai 3. Kemudian ada perintah insert 2, sehingga AVL Tree memiliki tiga node dengan nilai 3, 1, dan 2. Node dengan nilai 2 ditambahkan sebagai anak kiri dari node dengan nilai 1.

Setelah ketiga node ditambahkan, subtree kiri dari node dengan nilai 3 memiliki tinggi yang lebih rendah daripada subtree kanan. Sehingga, dilakukan rotasi kiri terhadap node dengan nilai 1 sebagai pivotnya. Setelah itu, dilakukan rotasi kanan terhadap node dengan nilai 3 sebagai pivotnya. Setelah dilakukan rotasi kiri dan rotasi kanan, struktur dari AVL Tree menjadi seperti yang ditunjukkan pada gambar di atas. Pada akhirnya, ada perintah preOrder yang meminta menampilkan data dari AVL Tree secara preOrder. Dengan preOrder, maka data yang ditampilkan adalah 2, 1, dan 3.