

LAPORAN PRAKTIKUM

“Post Test Pertemuan I”

Diajukan untuk memenuhi salah satu praktikum Mata Kuliah Matematika Diskrit yang di
ampu oleh:

Nur Rochmah Dyah PA, S.T., M.Kom.



Disusun Oleh:

Mohammad Farid Hendianto 2200018401

UNIVERSITAS AHMAD DAHLAN
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI INFORMATIKA
TAHUN 2023

Berikut adalah program untuk mencari Irisan, Gabungan, dan Modulus dari suatu himpunan

[illegible]

```

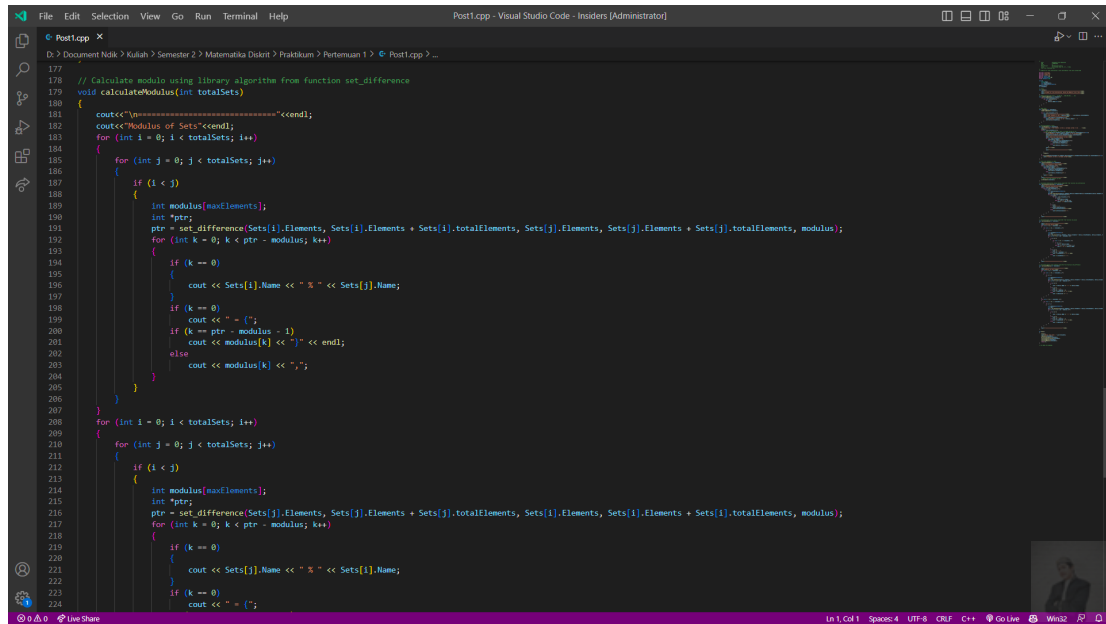
Post1.cpp - Visual Studio Code - Insiders [Administrator]
D:\Document Ndik > Kuliah > Semester 2 > Matematika Diskrit > Praktikum > Portemuan 1 > Post1.cpp > ...
107 // Calculate Intersection using library algorithm from function set_intersection
108 void calculateIntersection(int totalSets){
109     cout<<"\n-----"<<endl;
110     cout<<"Calculate Intersection"<<endl;
111     for(int i=0; i<totalSets; i++){
112         for(int j=i+1; j<totalSets; j++){
113             if (i < j){
114                 int intersection[maxElements];
115                 int *ptr;
116                 ptr = set_intersection(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, intersection);
117                 for (int k=0; k<ptr-intersection; k++){
118                     if (k==0){
119                         for (int l=0; l<totalSets; l++){
120                             if (l==i){
121                                 cout<<Sets[l].Name;
122                             } else if (l==j || l>1 && l<j){
123                                 cout<<" " <<Sets[l].Name;
124                             }
125                         }
126                     }
127                     if (k==ptr-intersection-1){
128                         cout<<" ";
129                     } else {
130                         cout<<intersection[k]<<" "<<endl;
131                     } else {
132                         cout<<intersection[k]<<" ";
133                     }
134                 }
135             }
136         }
137     }
138     cout<<"\n-----"<<endl;
139 }
140 // Calculate Union using library algorithm from function set_union
141 void calculateUnion(int totalSets){
142     cout<<"\n-----"<<endl;
143     cout<<"Calculate Union"<<endl;
144     for (int i = 0; i < totalSets; i++)
145     {
146         for (int j = 0; j < totalSets; j++)
147         {
148             if (i < j)
149             {
150                 int unionSet[maxElements];
151                 int *ptr;
152                 ptr = set_union(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, unionSet);
153                 for (int k = 0; k < ptr - unionSet; k++)
154                 {
155                     if (k == 0)
156                     {
157                         for (int l = 0; l < totalSets; l++)
158                         {
159                             if (l == i)
160                                 cout << Sets[l].Name;
161                             else if (l == j || l > 1 && l < j)
162                                 cout << " U " << Sets[l].Name;
163                         }
164                     }
165                     if (k == ptr - unionSet - 1)
166                         cout << " ";
167                     else {
168                         cout << unionSet[k] << " " << endl;
169                     } else {
170                         cout << unionSet[k] << " ";
171                     }
172                 }
173             }
174         }
175     }
176     cout<<"\n-----"<<endl;
177 }
178 // Calculate Modulo using library algorithm from function set_difference
179 void calculateModulus(int totalSets){
180     cout<<"\n-----"<<endl;
181     cout<<"Modulo of Sets"<<endl;
182     for (int i = 0; i < totalSets; i++)
183     {
184         for (int j = 0; j < totalSets; j++)
185         {
186             if (i < j)
187             {
188                 int differenceSet[maxElements];
189                 int *ptr;
190                 ptr = set_difference(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, differenceSet);
191                 for (int k = 0; k < ptr - differenceSet; k++)
192                 {
193                     if (k == 0)
194                     {
195                         for (int l = 0; l < totalSets; l++)
196                         {
197                             if (l == i)
198                                 cout << Sets[l].Name;
199                             else if (l == j || l > 1 && l < j)
200                                 cout << " D " << Sets[l].Name;
201                         }
202                     }
203                     if (k == ptr - differenceSet - 1)
204                         cout << " ";
205                     else {
206                         cout << differenceSet[k] << " " << endl;
207                     } else {
208                         cout << differenceSet[k] << " ";
209                     }
210                 }
211             }
212         }
213     }
214     cout<<"\n-----"<<endl;
215 }
216 }
217 int main()
218 {
219     int totalSets;
220     cout<<"Enter total sets: ";
221     cin>>totalSets;
222     while (totalSets < 1 || totalSets > 10)
223     {
224         cout<<"Invalid input. Please enter a value between 1 and 10: ";
225         cin>>totalSets;
226     }
227     int *Sets;
228     Sets = new int[totalSets * maxElements];
229     for (int i = 0; i < totalSets; i++)
230     {
231         int *Set;
232         Set = new int[maxElements];
233         for (int j = 0; j < maxElements; j++)
234             Set[j] = 0;
235         Sets[i * maxElements] = Set;
236         delete Set;
237     }
238     delete[] Sets;
239     cout<<"Enter elements for each set: ";
240     for (int i = 0; i < totalSets; i++)
241     {
242         cout<<"Set " << i << ": ";
243         for (int j = 0; j < maxElements; j++)
244             cout<<Set[i * maxElements + j] << " ";
245         cout<<"\n";
246     }
247     calculateIntersection(totalSets);
248     calculateUnion(totalSets);
249     calculateModulus(totalSets);
250     return 0;
251 }

```

```

Post1.cpp - Visual Studio Code - Insiders [Administrator]
D:\Document Ndik > Kuliah > Semester 2 > Matematika Diskrit > Praktikum > Portemuan 1 > Post1.cpp > ...
139 // Calculate Union using library algorithm from function set_union
140 void calculateUnion(int totalSets){
141     cout<<"\n-----"<<endl;
142     cout<<"Calculate Union"<<endl;
143     for (int i = 0; i < totalSets; i++)
144     {
145         for (int j = 0; j < totalSets; j++)
146         {
147             if (i < j)
148             {
149                 int unionSet[maxElements];
150                 int *ptr;
151                 ptr = set_union(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, unionSet);
152                 for (int k = 0; k < ptr - unionSet; k++)
153                 {
154                     if (k == 0)
155                     {
156                         for (int l = 0; l < totalSets; l++)
157                         {
158                             if (l == i)
159                                 cout << Sets[l].Name;
160                             else if (l == j || l > 1 && l < j)
161                                 cout << " U " << Sets[l].Name;
162                         }
163                     }
164                     if (k == ptr - unionSet - 1)
165                         cout << " ";
166                     else {
167                         cout << unionSet[k] << " " << endl;
168                     } else {
169                         cout << unionSet[k] << " ";
170                     }
171                 }
172             }
173         }
174     }
175     cout<<"\n-----"<<endl;
176 }
177 // Calculate Modulo using library algorithm from function set_difference
178 void calculateModulus(int totalSets){
179     cout<<"\n-----"<<endl;
180     cout<<"Modulo of Sets"<<endl;
181     for (int i = 0; i < totalSets; i++)
182     {
183         for (int j = 0; j < totalSets; j++)
184         {
185             if (i < j)
186             {
187                 int differenceSet[maxElements];
188                 int *ptr;
189                 ptr = set_difference(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, differenceSet);
190                 for (int k = 0; k < ptr - differenceSet; k++)
191                 {
192                     if (k == 0)
193                     {
194                         for (int l = 0; l < totalSets; l++)
195                         {
196                             if (l == i)
197                                 cout << Sets[l].Name;
198                             else if (l == j || l > 1 && l < j)
199                                 cout << " D " << Sets[l].Name;
200                         }
201                     }
202                     if (k == ptr - differenceSet - 1)
203                         cout << " ";
204                     else {
205                         cout << differenceSet[k] << " " << endl;
206                     } else {
207                         cout << differenceSet[k] << " ";
208                     }
209                 }
210             }
211         }
212     }
213     cout<<"\n-----"<<endl;
214 }
215 }
216 int main()
217 {
218     int totalSets;
219     cout<<"Enter total sets: ";
220     cin>>totalSets;
221     while (totalSets < 1 || totalSets > 10)
222     {
223         cout<<"Invalid input. Please enter a value between 1 and 10: ";
224         cin>>totalSets;
225     }
226     int *Sets;
227     Sets = new int[totalSets * maxElements];
228     for (int i = 0; i < totalSets; i++)
229     {
230         int *Set;
231         Set = new int[maxElements];
232         for (int j = 0; j < maxElements; j++)
233             Set[j] = 0;
234         Sets[i * maxElements] = Set;
235         delete Set;
236     }
237     delete[] Sets;
238     cout<<"Enter elements for each set: ";
239     for (int i = 0; i < totalSets; i++)
240     {
241         cout<<"Set " << i << ": ";
242         for (int j = 0; j < maxElements; j++)
243             cout<<Set[i * maxElements + j] << " ";
244         cout<<"\n";
245     }
246     calculateIntersection(totalSets);
247     calculateUnion(totalSets);
248     calculateModulus(totalSets);
249     return 0;
250 }

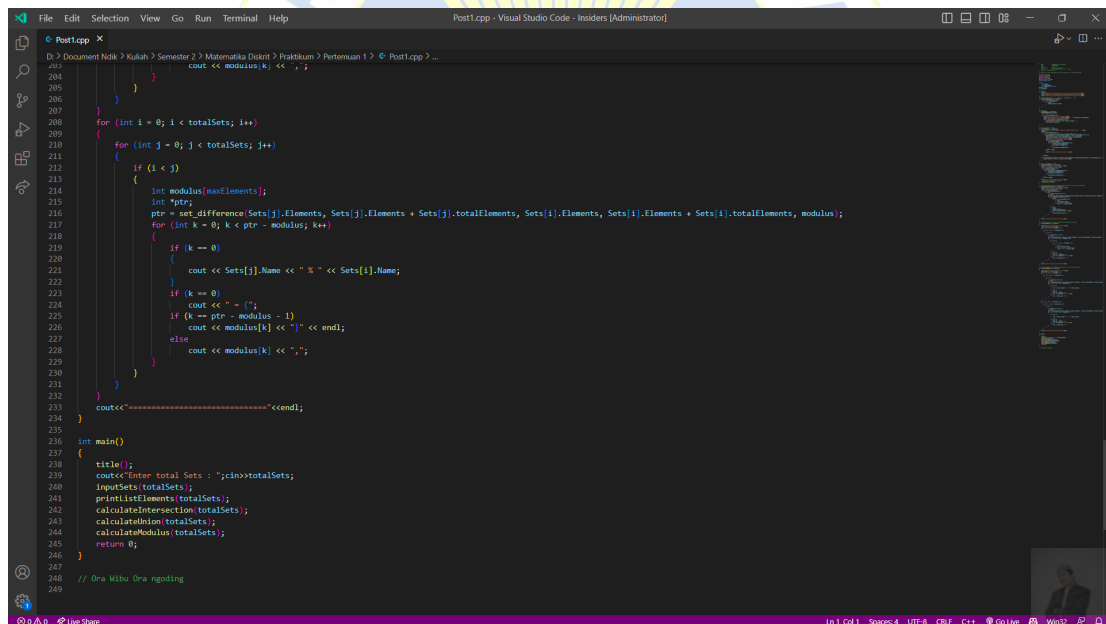
```



```

177 // Calculate modulo using library algorithm from function set_difference
178 void calculateModulus(int totalSets)
179 {
180     cout<<"Modulus of Sets"<<endl;
181     cout<<"Modulus of Sets"<<endl;
182     for (int i = 0; i < totalSets; i++)
183     {
184         for (int j = 0; j < totalSets; j++)
185         {
186             if (i < j)
187             {
188                 int modulus[maxElements];
189                 int *ptr;
190                 ptr = set_difference(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, modulus);
191                 for (int k = 0; k < ptr - modulus; k++)
192                 {
193                     if (k == 0)
194                     {
195                         cout << Sets[i].Name << " X " << Sets[j].Name;
196                     }
197                     if (k == 0)
198                     {
199                         cout << " = ";
200                         if (k == ptr - modulus - 1)
201                             cout << modulus[k] << " " << endl;
202                         else
203                             cout << modulus[k] << " ";
204                     }
205                 }
206             }
207         }
208     }
209     for (int i = 0; i < totalSets; i++)
210     {
211         for (int j = 0; j < totalSets; j++)
212         {
213             if (i < j)
214             {
215                 int modulus[maxElements];
216                 int *ptr;
217                 ptr = set_difference(Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, modulus);
218                 for (int k = 0; k < ptr - modulus; k++)
219                 {
220                     if (k == 0)
221                     {
222                         cout << Sets[j].Name << " X " << Sets[i].Name;
223                     }
224                     if (k == 0)
225                     {
226                         cout << " = ";
227                         if (k == ptr - modulus - 1)
228                             cout << modulus[k] << " " << endl;
229                         else
230                             cout << modulus[k] << " ";
231                     }
232                 }
233             }
234         }
235     }
236     cout<<"Modulus of Sets"<<endl;
237 }
238
239 int main()
240 {
241     title();
242     cout<<"Enter total Sets : ";cin>>totalSets;
243     inputSets(totalSets);
244     printInitialElements(totalSets);
245     calculateIntersection(totalSets);
246     calculateUnion(totalSets);
247     calculateModulus(totalSets);
248     return 0;
249 }
250
251 // Ora Wilu Ora ngoding
252

```



```

253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

// C++ program to find the sum of all prime numbers in a given range
// using Sieve of Eratosthenes
#include <iostream>
using namespace std;

// Function to find the sum of all prime numbers in a given range
// using Sieve of Eratosthenes
int sumOfPrimes(int n)
{
    // Create a boolean array "isPrime[0..n-1]" and initialize
    // all entries as true. A value at isPrime[i] will finally
    // be false if i is not a prime, else true.
    bool isPrime[n];
    for (int i = 0; i < n; i++)
        isPrime[i] = true;

    // Sieve of Eratosthenes
    for (int p = 2; p * p < n; p++)
    {
        // If isPrime[p] is not changed, then it is a prime
        if (isPrime[p])
        {
            // Update all multiples of p
            for (int i = p * p; i < n; i += p)
                isPrime[i] = false;
        }
    }

    // Sum of all prime numbers
    int sum = 0;
    for (int i = 2; i < n; i++)
    {
        if (isPrime[i])
            sum += i;
    }

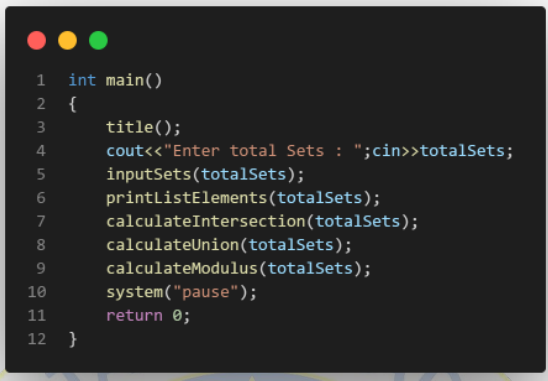
    return sum;
}

// Driver code
int main()
{
    int n = 100;
    cout << "Sum of all prime numbers in the range [2, 100] is: "
        << sumOfPrimes(n) << endl;

    return 0;
}

```

Kodingan ini merupakan program untuk mencari irisan, gabungan dan sisa bagi antara set menggunakan bahasa pemrograman C++. Program ini memiliki perhitungan kalku Berikut ini penjelasan secara lebih detail tentang setiap fungsi yang digunakan pada program ini:



```
1  int main()
2  {
3      title();
4      cout<<"Enter total Sets : ";cin>>totalSets;
5      inputSets(totalSets);
6      printListElements(totalSets);
7      calculateIntersection(totalSets);
8      calculateUnion(totalSets);
9      calculateModulus(totalSets);
10     system("pause");
11     return 0;
12 }
```

- title(): fungsi untuk menampilkan judul program pada layar.
- numberingNameSets(): fungsi untuk memberikan nama pada set dengan format A, B, C, AA, AB, AC, AAA, AAB, AAC, dst.
- inputSets(): fungsi untuk meminta pengguna memasukkan elemen dari set-set yang akan diproses. Setiap set akan diminta jumlah elemennya terlebih dahulu, kemudian pengguna diminta memasukkan elemen-elemennya satu persatu.
- sortElements(): fungsi untuk mengurutkan elemen-elemen pada set secara ascending. Fungsi ini menggunakan fungsi is_sorted() untuk memeriksa apakah elemen pada set sudah terurut atau belum. Jika belum, maka fungsi sort() akan digunakan untuk mengurutkannya. Setelah diurutkan, elemen-elemen pada set akan ditampilkan kembali pada layar.
- printListElements(): fungsi untuk menampilkan elemen-elemen pada set pada layar.
- calculateIntersection(): fungsi untuk menghitung irisan antara set-set yang ada menggunakan fungsi set_intersection() dari library algorithm pada C++. Fungsi ini akan menampilkan hasil irisan pada layar.
- calculateUnion(): fungsi untuk menghitung gabungan antara set-set yang ada menggunakan fungsi set_union() dari library algorithm pada C++. Fungsi ini akan menampilkan hasil gabungan pada layar.
- calculateModulus(): fungsi untuk menghitung sisa bagi antara set-set yang ada. Fungsi ini akan menampilkan hasil sisa bagi pada layar.

Selain fungsi-fungsi di atas, program ini juga menggunakan beberapa variabel global seperti Sets untuk menyimpan data set, totalSets untuk menyimpan jumlah set yang akan diproses, dan maxSets dan maxElements untuk menyimpan nilai maksimum dari jumlah set dan jumlah elemen pada set. Program ini juga menggunakan directive #include untuk

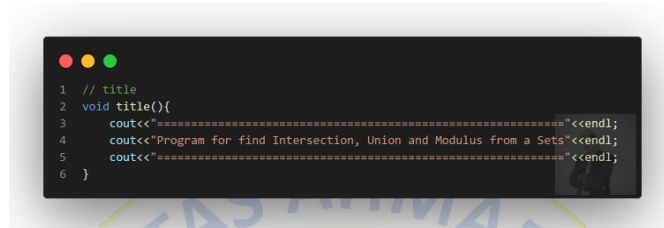
memasukkan library yang digunakan pada program dan using namespace std untuk mempermudah penggunaan istilah pada C++.



Kutipan kode tersebut merupakan sebuah program yang digunakan untuk mencari persimpangan (intersection), gabungan (union), dan modulus dari himpunan-himpunan. Program tersebut menggunakan beberapa library yaitu iostream, algorithm, dan unordered_set.

Selanjutnya, terdapat penggunaan definisi dari konstanta maxSets dan maxElements dengan nilai masing-masing 100, yang kemudian akan digunakan sebagai batas maksimum jumlah himpunan dan elemen yang dapat diproses oleh program tersebut.

Program tersebut juga menggunakan sebuah struktur data yang bernama Sets. Struktur data tersebut berisi beberapa variabel, yaitu Name, totalElements, dan Elements. Variabel Name berfungsi untuk menyimpan nama himpunan, totalElements untuk menyimpan jumlah elemen dalam himpunan tersebut, dan Elements untuk menyimpan nilai dari elemen-elemen yang ada dalam himpunan tersebut. Struktur data Sets digunakan untuk menyimpan informasi tentang himpunan-himpunan yang akan diproses oleh program. Selanjutnya, terdapat sebuah variabel totalSets yang digunakan untuk menyimpan jumlah himpunan yang akan diproses oleh program.



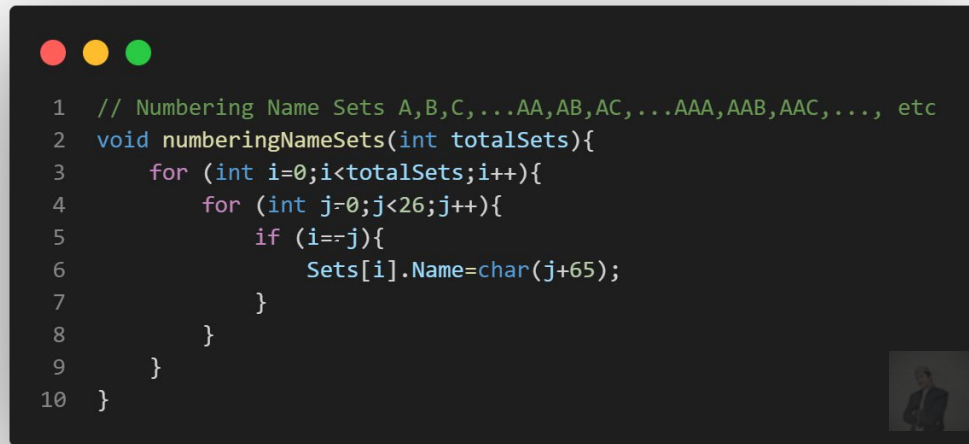
```
1 // title
2 void title(){
3     cout<<"===== "<<endl;
4     cout<<"Program for find Intersection, Union and Modulus from a Sets"<<endl;
5     cout<<"===== "<<endl;
6 }
```

Kode tersebut adalah sebuah fungsi dengan nama "title" yang berfungsi untuk menampilkan judul program pada console. Pada fungsi tersebut terdapat tiga statement output yang menggunakan fungsi "cout" yang berfungsi untuk menampilkan teks ke dalam console.

Teks yang ditampilkan pada fungsi "title" adalah sebagai berikut:

```
"===== "
"Program for find Intersection, Union and Modulus from a Sets"
"===== "
```

Dengan menampilkan judul program yang jelas dan deskriptif, pengguna program akan dapat memahami dengan lebih mudah tentang apa yang program tersebut lakukan dan bagaimana cara menggunakannya. Selain itu, pengguna juga dapat lebih mudah membedakan program tersebut dengan program lain yang serupa.

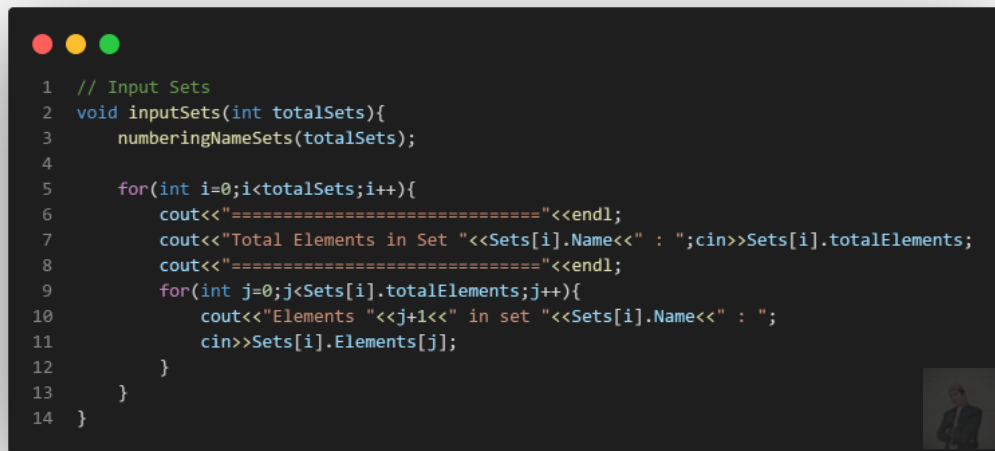


```
1 // Numbering Name Sets A,B,C,...AA,AB,AC,...AAA,AAB,AAC,..., etc
2 void numberingNameSets(int totalSets){
3     for (int i=0;i<totalSets;i++){
4         for (int j=0;j<26;j++){
5             if (i==j){
6                 Sets[i].Name=char(j+65);
7             }
8         }
9     }
10 }
```

Kode ini merupakan sebuah fungsi dengan nama "numberingNameSets" yang berfungsi untuk memberikan nomor atau label pada setiap himpunan yang ada. Fungsi ini memiliki satu parameter yaitu "totalSets" yang menentukan jumlah himpunan yang akan diberi nomor. Kode ini menggunakan nested loop atau perulangan bersarang, dimana perulangan pertama akan berjalan sebanyak "totalSets" kali, dan perulangan kedua akan berjalan sebanyak 26 kali.

Pada setiap iterasi perulangan pertama, fungsi akan memeriksa nomor indeks himpunan apakah sama dengan nomor indeks alfabet, jika i sama dengan j , maka fungsi akan memberikan label alfabet pada himpunan tersebut berdasarkan urutan alfabet. Sebagai contoh, jika $i=0$ dan $j=0$, maka nama himpunan tersebut akan diberi label "A", jika $i=1$ dan $j=0$, maka nama himpunan tersebut akan diberi label "B", dan seterusnya. Setiap nama himpunan yang diberikan label akan disimpan dalam variabel `Sets[i].Name`, dimana i adalah nomor indeks himpunan.

Dengan menggunakan kode ini, kita dapat memberikan label pada setiap himpunan secara otomatis, sehingga mempermudah identifikasi dan penggunaan himpunan tersebut dalam program. Kode ini juga dapat diubah atau dimodifikasi untuk memberikan label dengan format yang berbeda.



```

1 // Input Sets
2 void inputSets(int totalSets){
3     numberingNameSets(totalSets);
4
5     for(int i=0;i<totalSets;i++){
6         cout<<"======"<<endl;
7         cout<<"Total Elements in Set "<<Sets[i].Name<<" : ";cin>>Sets[i].totalElements;
8         cout<<"======"<<endl;
9         for(int j=0;j<Sets[i].totalElements;j++){
10             cout<<"Elements "<<j+1<<" in set "<<Sets[i].Name<<" : ";
11             cin>>Sets[i].Elements[j];
12         }
13     }
14 }

```

Kode yang diberikan adalah sebuah fungsi bernama "inputSets" yang memiliki satu parameter "totalSets" bertipe integer. Fungsi ini bertujuan untuk mengambil masukan dari pengguna mengenai himpunan (set) dan elemen-elemen yang terkandung di dalamnya.

Pertama-tama, fungsi akan memanggil fungsi "numberingNameSets" untuk memberikan nomor pada set. Kemudian, dengan menggunakan loop "for" sebanyak "totalSets" kali, program akan meminta input dari pengguna untuk jumlah elemen dalam set dan menyimpannya ke dalam variabel "totalElements". Setelah itu, program akan mengulangi proses untuk meminta input dari pengguna sebanyak "totalElements" kali untuk setiap elemen dalam set tersebut.

Kemudian, program akan menampilkan pesan "======" sebagai pemisah antar set, dan menampilkan nomor elemen dan nilai elemen yang dimasukkan oleh pengguna. Pesan ini akan ditampilkan ke layar dengan menggunakan fungsi "cout", sedangkan input dari pengguna akan diambil dengan menggunakan fungsi "cin" dan disimpan dalam variabel "Elements" yang merupakan bagian dari struct "Sets".

Dengan demikian, fungsi "inputSets" bertujuan untuk mengambil masukan dari pengguna mengenai himpunan dan elemen-elemen yang terkandung di dalamnya, dan menyimpannya dalam struktur data yang telah didefinisikan sebelumnya.

```

1  / Sorting Elements in Sets
2  void sortElements(int totalSets){
3      cout<<"Checking if the Elements in Sets is already sorted or not . . . "<<endl;
4      sleep(2);
5      for(int i=0;i<totalSets;i++){
6          // check if the Elements in Sets is already sorted or not
7          if (is_sorted(Sets[i].Elements,Sets[i].Elements+Sets[i].totalElements)==false){
8              cout<<"Elements in Set "<<Sets[i].Name<<" is not sorted"<<endl;
9              sort(Sets[i].Elements,Sets[i].Elements+Sets[i].totalElements);
10             cout<<"===== "<<endl;
11             cout<<"After Sort the Elements in Sets"<<endl;
12             for (int i=0;i<totalSets;i++){
13                 cout<<Sets[i].Name<<" = { ";
14                 for (int j=0;j<Sets[i].totalElements;j++){
15                     if (j==Sets[i].totalElements-1)
16                         cout<<Sets[i].Elements[j];
17                     else
18                         cout<<Sets[i].Elements[j]<<" ";
19                 }
20                 cout<<" "<<endl;
21             }
22             cout<<"===== "<<endl;
23         }
24         sleep(1);
25     }
26 }
27 if (is_sorted(Sets[totalSets-1].Elements,Sets[totalSets-1].Elements+Sets[totalSets-1].totalElements)==true){
28     cout<<"Elements in Sets is already sorted"<<endl;
29 }
30 }

```

Kode di atas merupakan sebuah fungsi yang bernama "sortElements" dengan satu parameter masukan yaitu "totalSets" yang merupakan jumlah keseluruhan himpunan pada program. Fungsi ini bertujuan untuk memeriksa apakah elemen-elemen yang terkandung dalam set sudah terurut atau belum. Untuk itu, program menggunakan loop "for" untuk mengecek setiap set pada program. Dalam loop, program akan memeriksa apakah elemen dalam set sudah terurut atau belum dengan menggunakan fungsi "is_sorted". Jika elemen dalam set belum terurut, program akan melakukan sorting elemen tersebut dengan menggunakan fungsi "sort". Setelah itu, program akan menampilkan pesan bahwa elemen dalam set tidak terurut dan menampilkan kembali elemen-elemen dalam set setelah dilakukan pengurutan.

Setelah mengecek semua himpunan, program akan memeriksa apakah elemen terakhir dalam program sudah terurut atau belum dengan menggunakan fungsi "is_sorted". Jika elemen terakhir sudah terurut, program akan menampilkan pesan bahwa elemen dalam set sudah terurut.

Dalam menampilkan elemen-elemen dalam set, program akan menampilkan nama set terlebih dahulu kemudian menampilkan semua elemen dalam set tersebut dengan menggunakan loop "for". Elemen yang terakhir tidak perlu diberikan koma sebagai pemisah antara elemen lainnya. Dalam kode tersebut, terdapat pula penggunaan fungsi "sleep" untuk memberikan jeda waktu pada program sebelum menampilkan pesan atau melakukan operasi tertentu. Fungsi ini membutuhkan argumen berupa jumlah detik untuk jeda.

```
1 // print list Elements in Sets
2 void printListElements(int totalSets){
3     cout<<"====="<<endl;
4     cout<<"List Elements in Sets"<<endl;
5     for(int i=0;i<totalSets;i++){
6         cout<<Sets[i].Name<<" = {";
7         for(int j=0;j<Sets[i].totalElements;j++){
8             if (j==Sets[i].totalElements-1)
9                 cout<<Sets[i].Elements[j];
10            else
11                cout<<Sets[i].Elements[j]<<",";
12        }
13        cout<<"}"<<endl;
14    }
15    cout<<"====="<<endl;
16    // after sort the Elements in Sets
17    sortElements(totalSets);
18 }
```

Kode ini berisi sebuah fungsi bernama `printListElements` yang akan mencetak elemen-elemen dalam set. Fungsi ini menerima satu argumen yaitu `totalSets` yang menyatakan jumlah set yang ingin dicetak.

Fungsi ini pertama-tama mencetak garis putus-putus dan judul "List Elements in Sets" untuk memulai outputnya. Kemudian, untuk setiap set, fungsi ini mencetak nama set diikuti dengan tanda sama dengan (=) dan kurung kurawal ({}). Selanjutnya, fungsi ini akan mengulangi elemen-elemen dalam set dengan menggunakan perulangan `for`. Untuk setiap elemen, fungsi ini akan mencetak elemen tersebut diikuti dengan tanda koma (,) kecuali elemen terakhir. Setelah mencetak semua elemen, fungsi ini akan menutup kurung kurawal (}) dan pindah ke baris baru.

Setelah selesai mencetak semua set, fungsi ini mencetak garis putus-putus lagi untuk menandakan bahwa output telah selesai. Terakhir, fungsi akan memanggil fungsi lain yang bernama `sortElements` untuk mengurutkan elemen-elemen dalam set.

Dalam keseluruhan, kode ini bertujuan untuk mencetak elemen-elemen dalam set secara terstruktur dan juga mengurutkan elemen-elemen tersebut. Fungsi ini terdiri dari beberapa perintah penggunaan output standar yang biasa digunakan dalam bahasa C++, seperti `cout` dan `endl`.

```

1 // Calculate Intersection using library algorithm from function set_intersection
2 void calculateIntersection(int totalSets){
3     cout<<"\n=====<endl;
4     cout<<"Calculate Intersection"<endl;
5     for(int i=0; i<totalSets; i++){
6         for(int j=0; j<totalSets; j++){
7             if (i<j){
8                 int intersection[maxElements];
9                 int *ptr;
10                ptr=set_intersection(Sets[i].Elements, Sets[i].Elements+Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements+Sets[j].totalElements, intersection);
11                for (int k=0; k<ptr-intersection; k++){
12                    if (k==0){
13                        for (int l=0; l<totalSets; l++){
14                            if (l==i)
15                                cout<<Sets[l].Name;
16                            else if (l==j || l>i && l<j)
17                                cout<<" " <<Sets[l].Name;
18                        }
19                    }
20                    if (k==0)
21                        cout<<" = {";
22                    if (k==ptr-intersection-1)
23                        cout<<intersection[k]<<" "<<endl;
24                    else
25                        cout<<intersection[k]<<",";
26                }
27            }
28        }
29    }
30    cout<<"\n=====<endl;
31 }

```

```

1 // Calculate Union using library algorithm from function set_union
2 void calculateUnion(int totalSets)
3 {
4     cout<<"\n=====<endl;
5     cout<<"Calculate Union"<endl;
6     for (int i = 0; i < totalSets; i++)
7     {
8         for (int j = 0; j < totalSets; j++)
9         {
10             if (i < j)
11             {
12                 int unionSet[maxElements];
13                 int *ptr;
14                 ptr = set_union(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, unionSet);
15                 for (int k = 0; k < ptr - unionSet; k++)
16                 {
17                     if (k == 0)
18                     {
19                         for (int l = 0; l < totalSets; l++)
20                         {
21                             if (l == i)
22                                 cout << Sets[l].Name;
23                             else if (l == j || l > i && l < j)
24                                 cout <<" " << Sets[l].Name;
25                         }
26                     }
27                     if (k == 0)
28                         cout <<" = {";
29                     if (k == ptr - unionSet - 1)
30                         cout << unionSet[k] <<" "<< endl;
31                     else
32                         cout << unionSet[k] << ",";
33                 }
34             }
35        }
36    }
37    cout<<"\n=====<endl;
38 }

```



```

1 // Calculate modulos using library algorithm from function set_difference
2 void calculateModulus(int totalSets)
3 {
4     cout<<"=====<endl;
5     cout<<"Modulus of Sets"<endl;
6     for (int i = 0; i < totalSets; i++)
7     {
8         for (int j = 0; j < totalSets; j++)
9         {
10             if (i < j)
11             {
12                 int modulus[totalElements];
13                 int *ptr;
14                 ptr = set_difference(Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, modulus);
15                 for (int k = 0; k < ptr - modulus; k++)
16                 {
17                     if (k == 0)
18                     {
19                         cout << Sets[i].Name << " X " << Sets[j].Name;
20                     }
21                     if (k == 0)
22                         cout << " = {";
23                     if (k == ptr - modulus - 1)
24                         cout << modulus[k] << " }" << endl;
25                     else
26                         cout << modulus[k] << " ";
27                 }
28             }
29         }
30     }
31     for (int i = 0; i < totalSets; i++)
32     {
33         for (int j = 0; j < totalSets; j++)
34         {
35             if (i < j)
36             {
37                 int modulus[totalElements];
38                 int *ptr;
39                 ptr = set_difference(Sets[j].Elements, Sets[j].Elements + Sets[j].totalElements, Sets[i].Elements, Sets[i].Elements + Sets[i].totalElements, modulus);
40                 for (int k = 0; k < ptr - modulus; k++)
41                 {
42                     if (k == 0)
43                     {
44                         cout << Sets[j].Name << " X " << Sets[i].Name;
45                     }
46                     if (k == 0)
47                         cout << " = {";
48                     if (k == ptr - modulus - 1)
49                         cout << modulus[k] << " }" << endl;
50                     else
51                         cout << modulus[k] << " ";
52                 }
53             }
54         }
55     }
56     cout<<"=====<endl;
57 }

```

Kode tersebut merupakan bagian dari program yang digunakan untuk menghitung operasi himpunan, yaitu intersection (irisan), union (gabungan), dan modulus (selisih) dari beberapa himpunan yang diinputkan oleh user. Kode tersebut bekerja dengan cara menggunakan tiga fungsi yang berbeda yang diambil dari pustaka library yaitu:

- `set_intersection` yang digunakan untuk menghitung irisan antara dua himpunan
- `set_union` yang digunakan untuk menghitung gabungan antara dua himpunan
- `set_difference` yang digunakan untuk menghitung selisih antara dua himpunan. Dari selisih ini bisa untuk dicari hasil modulus.

Ketiga fungsi tersebut menggunakan algoritma untuk menghitung operasi himpunan yang diimplementasikan dalam bahasa C++.

Setiap fungsi akan diiterasi untuk setiap pasangan himpunan yang berbeda, di mana pasangan himpunan tersebut akan digunakan untuk menghitung operasi himpunan yang dimaksud. Setelah itu, hasil dari operasi himpunan tersebut akan ditampilkan ke layar.

Setiap hasil operasi himpunan ditampilkan dalam bentuk himpunan dengan elemen yang dipisahkan oleh tanda koma dan diapit oleh kurung kurawal. Selain itu, setiap himpunan akan dilengkapi dengan nama himpunan yang dioperasikan sehingga pengguna dapat mengetahui dengan jelas hasil operasi himpunan yang ditampilkan di layar.

Contoh output hasil kodingan.

Contoh 1

```

PS C:\Users\Widi> cd "d:\Document\Nolik\Kuliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 1\" ; if ($?) { g++ Post1.cpp -o Post1 } ; if ($?) { .\Post1 }
=====
Program for Find Intersection, Union and Modulus from a Sets
Enter total Sets : 4
=====
Total Elements In Set A : 1
Elements 1 in set A : 2
=====
Total Elements In Set B : 2
Elements 1 in set B : 4
Elements 2 in set B : 5
=====
Total Elements In Set C : 2
Elements 1 in set C : 4
Elements 2 in set C : 5
=====
Total Elements In Set D : 5
Elements 1 in set D : 3
Elements 2 in set D : 7
Elements 3 in set D : 2
Elements 4 in set D : 3
Elements 5 in set D : 4
=====
List Elements in Sets
A = {2}
B = {4,5}
C = {4,5}
D = {3,7,2,3,4}
=====
Checking if the Elements in Sets is already sorted or not . . .
Elements in Set D is not sorted
=====
After Sort the Elements in Sets
A = {2}
B = {4,5}
C = {4,5}
D = {2,3,3,4,7}
=====
Elements in Sets is already sorted

=====
Calculate Intersection
A n B n C n D = {2}
B n C = {4,5}
B n C n D = {4}
C n D = {4}
=====

=====
Calculate Union
A u B = {2,4,5}
A u B u C = {2,4,5}
A u B u C u D = {2,3,3,4,7}
B u C = {4,5}
B u C u D = {2,3,3,4,5,7}
C u D = {2,3,3,4,5,7}
=====

=====
Modulus of Sets
A x B = {2}
A x C = {2}
B x D = {5}
C x D = {5}
B x A = {4,5}
C x A = {4,5}
D x A = {3,3,4,7}
D x B = {2,3,5,7}
D x C = {2,3,5,7}
=====
  
```

```

=====
Elements 1 in set D : 3
Elements 2 in set D : 7
Elements 3 in set D : 2
Elements 4 in set D : 3
Elements 5 in set D : 4
=====
List Elements in Sets
A = {2}
B = {4,5}
C = {4,5}
D = {3,7,2,3,4}
=====
Checking if the Elements in Sets is already sorted or not . . .
Elements in Set D is not sorted
=====
After Sort the Elements in Sets
A = {2}
B = {4,5}
C = {4,5}
D = {2,3,3,4,7}
=====
Elements in Sets is already sorted

=====
Calculate Intersection
A n B n C n D = {2}
B n C = {4,5}
B n C n D = {4}
C n D = {4}
=====

=====
Calculate Union
A u B = {2,4,5}
A u B u C = {2,4,5}
A u B u C u D = {2,3,3,4,7}
B u C = {4,5}
B u C u D = {2,3,3,4,5,7}
C u D = {2,3,3,4,5,7}
=====

=====
Modulus of Sets
A x B = {2}
A x C = {2}
B x D = {5}
C x D = {5}
B x A = {4,5}
C x A = {4,5}
D x A = {3,3,4,7}
D x B = {2,3,5,7}
D x C = {2,3,5,7}
=====
  
```

Contoh 2

```

PS C:\Users\Ndik\ > cd "D:\Document Ndik\Kuliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 1\" ; if ($?) { g++ Post1.cpp -o Post1 } ; if ($?) { .\Post1 }
-----
Program for find Intersection, Union and Modulus from a Sets
-----
Enter total Sets : 2
-----
Total Elements in Set A : 5
-----
Elements 1 in set A : 1
Elements 2 in set A : 7
Elements 3 in set A : 3
Elements 4 in set A : 4
Elements 5 in set A : 5
-----
Total Elements in Set B : 5
-----
Elements 1 in set B : 6
Elements 2 in set B : 2
Elements 3 in set B : 2
Elements 4 in set B : 4
Elements 5 in set B : 6
-----
List Elements in Sets
A = {1,7,3,4,5}
B = {6,2,2,4,6}
-----
Checking if the Elements in Sets is already sorted or not . . .
Elements in Set A is not sorted
-----
After Sort the Elements in Sets
A = {1,3,4,5,7}
B = {2,2,4,6,6}
-----
Elements in Set B is not sorted
-----
After Sort the Elements in Sets
A = {1,3,4,5,7}
B = {2,2,4,6,6}
-----
Elements in Sets is already sorted
-----
Calculate Intersection
A ∩ B = {4}
-----
Calculate Union
A ∪ B = {1,2,2,3,4,5,6,6,7}
-----
Modulus of Sets
A Δ B = {1,3,5,7}
-----

```

```

-----
Enter total Sets : 2
-----
Total Elements in Set A : 5
-----
Elements 1 in set A : 1
Elements 2 in set A : 7
Elements 3 in set A : 3
Elements 4 in set A : 4
Elements 5 in set A : 5
-----
Total Elements in Set B : 5
-----
Elements 1 in set B : 6
Elements 2 in set B : 2
Elements 3 in set B : 2
Elements 4 in set B : 4
Elements 5 in set B : 6
-----
List Elements in Sets
A = {1,7,3,4,5}
B = {6,2,2,4,6}
-----
Checking if the Elements in Sets is already sorted or not . . .
Elements in Set A is not sorted
-----
After Sort the Elements in Sets
A = {1,3,4,5,7}
B = {2,2,4,6,6}
-----
Elements in Set B is not sorted
-----
After Sort the Elements in Sets
A = {1,3,4,5,7}
B = {2,2,4,6,6}
-----
Elements in Sets is already sorted
-----
Calculate Intersection
A ∩ B = {4}
-----
Calculate Union
A ∪ B = {1,2,2,3,4,5,6,6,7}
-----
Modulus of Sets
A Δ B = {1,3,5,7}
B Δ A = {2,2,6,6}
-----
PS D:\Document Ndik\Kuliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 1>

```


Contoh 3

```

PS C:\Users\Widlo> cd "C:\Document Belk\Kulliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 1\" ; if ($?) { g++ Post1.cpp -o Post1 } ; if ($?) { .\Post1 }

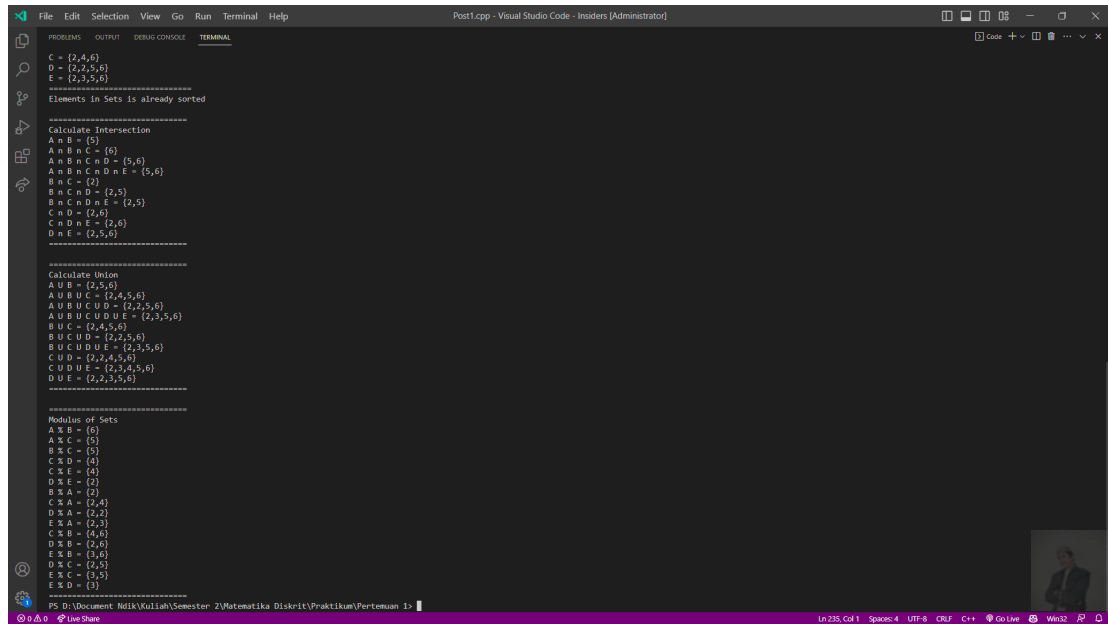
Program for find Intersection, Union and Modulus from a Sets
Enter total Sets : 5
=====
Total Elements in Set A : 2
=====
Elements 1 in set A : 5
Elements 2 in set A : 6
=====
Total Elements in Set B : 2
=====
Elements 1 in set B : 2
Elements 2 in set B : 5
=====
Total Elements in Set C : 3
=====
Elements 1 in set C : 6
Elements 2 in set C : 2
Elements 3 in set C : 4
=====
Total Elements in Set D : 4
=====
Elements 1 in set D : 5
Elements 2 in set D : 6
Elements 3 in set D : 2
Elements 4 in set D : 2
=====
Total Elements in Set E : 4
=====
Elements 1 in set E : 5
Elements 2 in set E : 6
Elements 3 in set E : 2
Elements 4 in set E : 3
=====
List Elements in Sets
A = {5,6}
B = {2,5}
C = {6,2,4}
D = {5,6,2,2}
E = {5,6,2,3}
=====
Checking if the Elements in Sets is already sorted or not . . .
Elements in Set C is not sorted
=====
After Sort the Elements in Sets
A = {5,6}
B = {2,5}
C = {2,4,6}
D = {5,6,2,2}
E = {5,6,2,3}
=====
Elements in Set D is not sorted

```

```

Checking if the Elements in Sets is already sorted or not . . .
Elements in Set C is not sorted
=====
After Sort the Elements in Sets
A = {5,6}
B = {2,5}
C = {2,4,6}
D = {5,6,2,2}
E = {5,6,2,3}
=====
Elements in Set D is not sorted
=====
After Sort the Elements in Sets
A = {5,6}
B = {2,5}
C = {2,4,6}
D = {2,2,5,6}
E = {5,6,2,3}
=====
Elements in Set E is not sorted
=====
After Sort the Elements in Sets
A = {5,6}
B = {2,5}
C = {2,4,6}
D = {2,2,5,6}
E = {2,3,5,6}
=====
Elements in Sets is already sorted
=====
Calculate Intersection
A n B = {5}
A n B n C = {6}
A n B n C n D = {5,6}
A n B n C n D n E = {5,6}
B n C = {2}
B n C n D = {2,5}
B n C n D n E = {2,5}
C n D = {2,6}
C n D n E = {2,6}
D n E = {2,5,6}
=====
Calculate Union
A u B = {2,5,6}
A u B u C = {2,4,5,6}
A u B u C u D = {2,2,5,6}
A u B u C u D u E = {2,3,5,6}
B u C = {2,4,5,6}
B u C u D = {2,2,5,6}
B u C u D u E = {2,3,5,6}
C u D = {2,2,4,5,6}

```



```

C = {2,4,6}
D = {2,3,5,6}
E = {2,3,5,6}
=====
Elements in Sets is already sorted
=====
Calculate Intersection
A n B = {5}
A n B n C = {6}
A n B n C n D = {5,6}
A n B n C n D n E = {5,6}
B n C = {2}
B n C n D = {2,5}
B n C n D n E = {2,5}
C n D = {2,5}
C n D n E = {2,6}
D n E = {2,5,6}
=====
Calculate Union
A u B = {2,5,6}
A u B u C = {2,4,5,6}
A u B u C u D = {2,2,5,6}
A u B u C u D u E = {2,3,5,6}
B u C = {2,4,5,6}
B u C u D = {2,2,5,6}
B u C u D u E = {2,3,5,6}
C u D = {2,2,4,5,6}
C u D u E = {2,3,4,5,6}
D u E = {2,2,3,5,6}
=====
Modulus of Sets
A % B = {6}
A % C = {5}
B % C = {5}
C % D = {4}
C % E = {4}
D % E = {2}
B % A = {2}
C % A = {2,4}
D % A = {2,2}
E % A = {2,3}
C % B = {4,6}
D % B = {2,6}
E % B = {3,6}
D % C = {2,5}
E % C = {3,5}
E % D = {3}
=====
PS D:\Document\Ndik\Kuliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 1>

```

Untuk full source dan program yang sudah di compile, dapat mengakses link github berikut.

<https://github.com/IRedDragonICY/Matematika-Diskrit>

