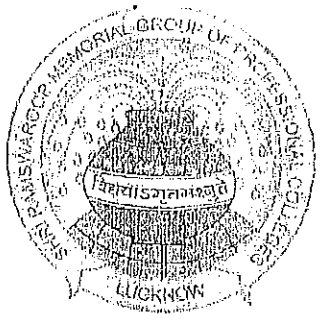


LAB WORK BOOK

For

DISCRETE STRUCTURES&LOGIC LAB

(KCS-353)

Course: B.Tech.

Branch: CS E Year: 2nd Semester: 3rd

Name of Student :	Amit Pandey		
College Roll No. :	BE18CS139	University Roll No. :	
Semester :	3rd	Session:	2019-20
Branch :	CSE	Group :	34
Lab Day :	Wednesday	Lab Period :	2:40 pm to 4:30 pm.

Progress Sheet

S.No.	Experiment	Date Of Issue	Date Exp. Done	Date Exp. Checked	Marks Obtained	Authorized Signatory
1	SHEET 01	14/08/19	14/08/19	21/8	3+2	
2	SHEET 02	21/08/19	21/08/19	4/9	3+2	
3	SHEET 03	04/09/19	04/09/19	11/09/19	3+2	
4	SHEET 04	11/09/19	11/09/19	18/09/19	3+2	
5	SHEET 05	18/09/19	18/09/19	25/09/19	3+2	
6	SHEET 06	25/09/19	25/09/19	9/10/19	3+2	
7	SHEET 07	16/10/19	16/10/19	23/10/19	3+1	
8	SHEET 08	23/10/19	23/10/19	06/11/19	3+1	
9	SHEET 09	13/11/19	13/11/19	13/11/19	3+2	
10	SHEET 10	13/11/19	13/11/19	13/11/19	3+2	

LAB-1

* Write a C program to perform intersection of two sets.

```
#include<stdio.h>
int main()
{
 int a[10],b[10],flag=0,n1,n2,j,i;
 printf("Enth Enter the array1 size:");
 scanf("%d",&n1);
 printf("Enter the array2 size:");
 scanf("%d",&n2);
 printf("Enter the array1 elements:");
 for(i=0;i<n1;i++)
  scanf("%d",a[i]);
 printf("Enter the array2 elements:");
 for(i=0;i<n2;i++)
  scanf("%d",&b[i]);
 printf("Intersection:");
 for(i=0;i<n1;i++)
 {
  for(j=0;j<n2;j++)
  {
   if(b[i]==a[j])
   {
    flag=1;
} }
```

```
if (flag == 1)
{
  printf("%d", b[i]);
}
  flag = 0;
}
return 0;
}
```

* Output *

```
Enter array1 size : 3
Enter array2 size : 3
Enter array1 elements : 1  2  3
Enter array2 elements : 2  3  4
Intersection : 2 3
```

LAB - 2

* C program to find Complement of a given set:

```
#include<stdio.h>
int main()
{
 int u[50], a[30], i,j, b[50], m, n, c=0, flag;
 printf("Enter the no. of elements of universal set:");
 scanf("%d",&n);
printf("Enter elements:");
   for(i=0; i<n; i++)
   scanf("%d",&u[i]);
 printf("Enter the elements of set");
  scanf("%d",&m);
printf("Enter elements:");
   for(i=0; i<m; i++)
    scanf("%d", &a[i]);
    for(i=0; i<n; i++)
    {  flag=0
     for(j=0; j<m; j++)
      {  if(u[i]==a[j])
        flag=1;
      }
      if(flag==0)
     {  b[c]=u[i];
        c++
     }
}
```

```
printf("Complement of set A is :\n");
    for (i=0; i<x; i++)
    printf("%d ", b[i]);
    return 0;
}
```

* Output *

Enter the number of elements in Universal set : 5

Enter the elements : 2 4 6 8 10

Enter elements of sets : 3

Enter element : 1 2 10

Complement of set A is :

4 6 8

* C program to find the difference between two sets.

```
#include <stdio.h>
int main()
{
 int a[10], b[10], c[10], j, i, flag=0, k=0, n, m;
printf("Enter no. of elements in set A:");
 scanf("%d", &n);
printf("Enter elements in set A:");
 for(i=0; i<n; i++)
 scanf("%d", &a[i]);
printf("Enter the no. of element in set B:");
 scanf("%d", &m);
printf("Enter the elements of set B:");
 for(i=0; i<m; i++)
 {
  flag=0;
  for(j=0; j<n; j++)
  {
   if(a[i]==b[j])
   flag=1;
  }
  if(flag==0)
  {
   c[k]=a[i];
   k++;
  }
}
```

```
printf("The difference is : {");
  for(i=0; i<x; i++)
  printf("%d", c[i]);
  printf("}");
  return 0;
}
```

* Output *

Enter no. of elements in set A : 4

Enter element in set A : 20 40 75 50

Enter no. of element in set B: 5

Enter elements in set B: 11 19 20 75 80

Difference is : {40, 50}

LAB - 4

* Write a C program to find symmetric difference of two sets.

```
#include<stdio.h>
int main()
{
 int a[20], b[20], i, size, size1, j, f=0, c=0;
  printf("Enter size of A array \n");
  scanf("%d", &size);
  printf("Enter set A \n");
    for(i=0; i<size; i++)
     {
       printf("Enter %d elements \n", i+1);
       scanf("%d", &a[i]);
     }
  printf("Enter size of B Array \n");
   scanf("%d", size1);
  printf("Enter set B \n");
    for(i=0; i<size1; i++)
     {
       printf("enter %d element \n", i+1);
        scanf("%d", &b[i]);
     }
 printf("Symmetric difference of the sets are \n");
      for(i=0; i<size1; i++)
      {  f=0;
         for(j=0; j<size; j++)
            {
```

```
if(a[i]==b[j]
    {
      f=1;
       break;
    }
  }
 if(f==0)
 {
  printf("%d  ", a[i]);
     c++;
  }
 }
for(i=0;i<size;i++)
  { f=0;
    for(j=0;j<size1;j++)
       {if(b[i]==a[j])
       { f=1;
         break;
       }
     }
     if(f==0)
      { printf("%d  ", b[i]);
        c++;
      }
   }
     if(c==0)
       printf("NULL SET");
   }
```

* Output *

Enter size of A array : 4
Enter the elements of set A:
Enter element 1 : 12
Enter element 2: 34
Enter element 3: 56
Enter element 4: 67

Enter size of B Array : 4
Enter the elements of Set B :
Enter element 1: 12
Enter element 2: 56
Enter element 3: 78
Enter element 4: 90

Symmetric difference of these sets :-
34 , 67 , 78 , 90

(B)

~~LAB-8~~

// C program to implement the logic gates AND, OR & NOT.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int a, b, ch;
  while(1)
  {
    printf("Enter your choice \n 1. AND \n 2. OR
                \n 3. NOT \n 4. EXIT \n");
    scanf("%d", &ch);
    switch(ch)
  {
    Case 1: printf("Enter the value of A and B:\n");
            scanf("%d %d", &A, &B);
            if(A==1 && B==1)
            printf("1");
            else
             printf("0");
            break;
    Case 2: printf("Enter A and B \n");
            scanf("%d %d", &A, &B);
            if(A==1 || B==1)
```

```
printf("1");
else
printf("0");
break;
Case 3: printf("Enter the value of A \n");
        scanf("%d", &A);
        if(A==0)
        printf("1");
      else
        printf("0");
      break;
Case 4: exit(0);
default: printf("Wrong choice !!! \n");
   }
}
  return 0;
}
```

* Enter the value of A and B : 1 , 0

0

Enter the value of A and B : 1 , 1

1

* Enter the value of A and B : 1 , 1

1

Enter the value of A and B : 0, 0

0

* Enter the value of A : 1

0

Enter the value of A : 0

1

(3)

LAB-5

Some basic commands in Prolog

1. Union ([1,2,3],[4],X).

   Stores the union in X

   ⇒ X = [1,2,3,4]

2. Subset([Set1],[Set2]).

   Returns true if set1 is subset of Set 2 else false

   ⇒ Subset([1,2,3],[1,2,3,4]).

   ⇒ true

3. Subtract([Set1],[Set2],Variable).

   Subtracts set2 from Set1 and stores in Variable

   ⇒ Subtract([1,2,3],[1,2],X).

   ⇒ X = [3]

4. length([Set1], <variable>).

   Stores length of set in variable.

   ⇒ length([a,b,c,d,e],X).

   ⇒ X = 5.

5. ~~Insertion~~ Intersection([Set1],[Set2],<variable>).

   Stores intersection of Set1 and Set2 in variable

   ⇒ intersection([a,b,c,d],[a,e,c,f],X).

   ⇒ X = [a,c]

6. member (Variable, [Set]).

Assign values of element in set in variable

⇒ member (X, [1, 2, 3]).

⇒ X = 1
⇒ X = 2
⇒ X = 3

7. Sum_list ([Set], <variable>).

⇒ Stores sum of elements of set and stores in variable

⇒ sum_list ([1, 2, 3, 4, 5), X).

⇒ X = 15

8. write ("text").

Prints the quoted text

⇒ write ("Hello world").

⇒ Hello World.

9. plus (digit1, digit2, ..., variable).

Stores sum of digit in variable

⇒ plus (2, 3, X).

⇒ X = 5

⇒ plus (20, 31, X).

⇒ X = 51

10. between (<lowerbound>, <upperbound>, <variable>).

runs loop from lower to upper bound & stores in variab

⇒ between (1, 3, X)

⇒ X = 1;
⇒ X = 2;
⇒ X = 3;

LAB-6

Family Tree using Prolog

```
father (ram, dinesh).
father (ram, suresh).
father (dinesh, rahul).
father (dinesh, rakesh).
grandfa (A,C) :- father (A,B), father (B,C).
siblings (Y,Z) :- father (X,Y), father (X,Z).
cousins (D,E) :- father (F,D), father (G,F), father (G,H),
                 father (H,E).
```

* Output *

```
?- siblings (I, Suresh).
I = dinesh;
I = suresh;

?- Cousins (I, Rajesh).
I = Rahul;
I = Rakesh;
I = Rajesh;
?- Cousins (rajesh, I).
I = rahul
I = rakesh
I = rajesh
```

? grandpa (ram, X).

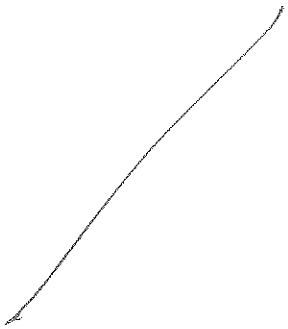
X = rahul ;
X = rakesh ;
X = rgesh.

? Siblings (rahul, X).

X = rahul ;
X = rakesh,

? Siblings (X, Suresh).

X = dinesh ;



* factorial using prolog.

```
fact(0,1).
fact(A,B):-
  A>0
  C is A-1,
  fact(C,D),
  B is A*D.
```

* Output →

```
fact(5,X).
  X = 120
```

* LAB-7 *

* Various operations in prolog

1. Union →

```
union([],[],[]).
union([H|T],[],[H|T]).
union([],[H|T],[H|T]).
union([H|T], S2, Result):-
        member(H,S2),
        union(T,S2,Result).
union([H|T],S2,[H|Result]):-
        not(member(H,S2)),
        union(T,S2,Result).
```

2. Intersection →

```
inter([],[],[]).
inter([],S2,[]).
inter([H|T],S2,[H|R]):-
        member(H,S2), inter(T,S2,R).
inter([_|T],S2,R):-
        inter(T,S2,R).
```

3. Subset →

```
subset([],[]).
subset([X|L],[X|S]):-
        subset(L,S).
subset(L,[_|S]):-
        subset(L,S).
```

* Output *

? union ([1,2,3], [2,3,4], X).

X = [1,2,3,4].

? union ([1,2], [], X).

X = [1,2].

? union ([], [13,14], X).

X = [13,14].

? inter ([1,2,3], [2,3], X).

X = [2,3].

? inter ([2,3,4], [], X).

X = [].

? subset ([1,2], [1,2,3]).

true.

? subset ([], [1]).

true.

? subset ([2,3,4], [1,2,3]).

false.

* LAB - Q*

- Write a function to find count of a particular number in a set also write a function to find its length.

1. len([],0).
   len([_|T],N):-
   len(T,M), N is M+1.

   Output →
   ?- len([1,2,3,4],X).
   X = 4.

2. count(X,[],0).
   count(X,[X|T],N):-
   count(X,T,M), N is M+1.
   count(X,[Y|T],N):-
   X\=Y, count(X,T,N).

   Output →
   ?- count(1,[1,2,1,1,3],X).
   X = 3.

- Write a function in prolog to find whether a number is odd or even, also write a function to print table.

```
Oe(1):-
write("odd").
Oe(0):-
write("even").
Oe(X):-
X1 is mod[X,2], Oe(X1).
```

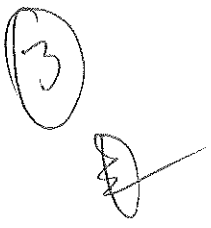
```
table(_,X,X).
table(N,0,X):-
0<X, Y is N*0, write(N*0), write("="),
write(Y), write("\n"), 0, is 0+1, table(N,0,,X).
```

Output →

```
?- oe(21)
   odd
   true.

?- table(7,1,1,1)
   7*1 = 7
   7*2 = 14
   7*3 = 21
   7*4 = 28
   7*5 = 35
   7*6 = 42
   7*7 = 49
   7*8 = 56
   7*9 = 63
   7*10 = 70

   true.
```



LAB-9

// C program to find the shortest distance. (Dijkstra).

```
#include<stdio.h>
#include<conio.h>
#define MAX 100
void dijkstra(int G[MAX][MAX], int n, int startnode)
int main()
{ int G[MAX][MAX], i,j, n, u;
  printf("Enter no. of vertex:");
  scanf("%d", &n)
  printf(" Enter adjacency matrix :\n");
      for(i=0; i<n; i++)
          for(j=0; j<n; j++)
              scanf("%d", &G[i][j]);
printf(" Enter the starting node");
  scanf("%d", &u);
    dijkstra(G, n, u);
    return 0;
}
void dikjstra(int G[max][MAX], int n, int startnode)
{
   int cost[MAX][MAX], distance[MAX], pred[MAX];
   int visited[MAX], count, mid_dis, next_node, i);
  for(i=0; i<n; i++)
      for(j=0; j<n; j++)
          if(G[i][j]==0)
              cost[i][j] = infinity;
          else
              cost[i][j] = G[i][j]
  for(i=0; i<n; i++)
      {  distance[i] = cost[startnode][i]
         pred[i] = startnode;
         visited[i] = 0;
```

```
distance[startnode]=0;
 visited[i]= startnode;
   visited[i] =0; }
while (count <n-1)
   {
     mis_distance = infinity;
      for(i=0; i<n; i++)
    if(distance[i] <mid_dis && !visited[i]);
      { mid_dis = dis[i];
          nextnode = i;
       }
   visited[next node]=1;
   for (i=0; i<n; i++)
       if (!visited[i])
     if (middis + cost[nextnode][i] <distance[i]
      { distance[i] = mid dist cost[next node][i];
          pred[i]=next node;
        }
       count ++;
       for(i=0; i<n; i++)
            if (i!=startnode)
          {
            printf("\n Distance of node %d = %d", i, distance[i]);
            printf("\n Path = %d", i);
            j= i;
           do
              {
                j=pred[j];
                printf(" <- %d", j);
               } while (j!=startnode);
          }
      }
```

* Output →

Enter the no. of vertices: 5

Enter the adjacency matrix:

0	10	0	30	100
10	0	50	0	0
0	50	0	20	10
30	0	20	0	10
100	0	10	60	0

Enter the starting node: 0

Distance of node 1 = 100

path = 1 <- 0

Distance of node 2 = 50

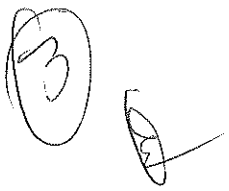
path = 2 <- 3 <- 0

Distance of node 3 = 30

path = 3 <- 0

Distance of node 4 = 60

path = 4 <- 2 <- 3 <- 0



LAB-10

Write a C program to perform union of two sets or array.

```
#include <stdio.h>
int main()
{
int a[10], b[10], flag=0, n2, n1, i, j;
printf("Enter array1 size:");
scanf("%d", &n1);
printf("Enter array2 size:");
scanf("%d", &n2);
printf("\n Enter array1 elements:");
for(i=0; i<n1; i++)
scanf("%d", a[i]);
printf("\n Enter array2 elements:");
for(i=0; i<n2; i++)
scanf("%d", &b[i]);
printf("\n Union is:");
for(i=0; i<n2; i++)
{
for(j=0; j<n1; j++)
{
if(b[i]==a[j]
{flag=1;
}
}
if(flag==0)
{
printf("%d", b[i]);
}flag=0;}
return 0;}
```

* Output *

Enter array1 size : 3

Enter array2 size : 3

Enter array1 elements : 1 2 3

Enter array2 elements : 4 5 6

Union : 1 2 3 4 5 6