Jawaharlal Nehru Engineering College

Laboratory Manual

DAA

For

S.Y (M.C.A) Students

FORWARD

It is my great pleasure to present this laboratory manual for S.Y (M.C.A) students for the subject of DAA.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9000 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9000 Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Prof. Dr.H.H.Shinde
Principal

LABORATORY MANUAL CONTENTS

This manual is intended for the students S.Y (M.C.A) engineering for the subject of DAA. Subject name itself is expecting what are the advance tools that might be Visual Studio to develop and build rich .NET applications. In the subject of this manual typically contains practical/Lab Sessions we have dealt with DAA.

Students are advised to thoroughly go through this manual rather than only topics mentioned are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Prof. G.R.Agrawal

Institute Vision

To create self-reliant, continuous learner & competent technocrats imbued with human values.

Institute Mission

- Imparting quality technical education to the students through participative teaching – learning process.
- Developing competence amongst the students through academic learning and practical experimentation.
- Inculcating social mindset and human values amongst the students.

Department Vision

Build a strong technical teaching and learning environment that responds swiftly to the challenges and needs of the current industry trends.

Department Mission

1. Provide excellent post graduate education in a state-of-the-art environment, preparing students for careers as computer technologist in self employment, industry, government and of IT enabled sectors.
2. Support society by participating in and encouraging technology transfer.

DOs and DON'T DOs in Laborary:

1. Do not handle any equipment before reading the instructions/Instruction manuals

2. Read carefully the power ratings of the equipment before it is switched on whether ratings 230 V/50
Hz or 115V/60 Hz. For Indian equipments, the power ratings are normally 230V/50Hz. If you have equipment with 115/60 Hz ratings, do not insert power plug, as our normal supply is 230V/50 Hz, which will damage the equipment.

3. Observe type of sockets of equipment power to avoid mechanical damage

4. Do not forcefully place connectors to avoid the damage

5. Strictly observe the instructions given by the teacher/Lab Instructor

Instruction for Laboratory Teachers::

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.

2. Students should be taught for taking the printouts under the observation of lab teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

LAB 1:- IMLPEMENTATION OF STACK AND QUEUE

```
//stack implementation.
#include<iostream.h>
#include<conio.h>
#include<dos.h>
void main()
{
        int arr[5];
        clrscr();
```

```
for(int i=0;i<5;i++)
arr[i]=0;
int ch;
do
{
cout<<"\n1:-push";
cout<<"\n2:-pop";
cout<<"\n3:-display";
cout<<"\n4:exit";
cout<<"\nenter choice";
cin>>ch;
switch(ch)
{
        case 1: int no,flag=0;
                cout<<"enter element" ;
                cin>>no;
                for(int i=0;i<5;i++)
                {
                        if(arr[i]==0)
                        {
                                arr[i]=no;
                                flag=1;
                                break;
                        }

                }
                if(flag==0)
                cout<<"stack is full";
                break;

        case 2: flag=0;
                for(i=4;i>=0;i--)
                {
                        if(arr[i]!=0)
                        {       cout<<arr[i];
                                arr[i]=0;
                                flag=1;
                                break;
                        }

                }
                if(flag==0)
                cout<<"stack is empty";
                break;

        case 3:for(i=0;i<5;i++)
```

```
                cout<<arr[i];
                break;
        //   case 4:exit(0);

        }
        }while(ch!=4);
}



//stack implementation.
#include<iostream.h>
#include<conio.h>
#include<dos.h>
void main()
{
        int arr[5];
        clrscr();
        for(int i=0;i<5;i++)
        arr[i]=0;
        int ch;
        do
        {
        cout<<"\n1:-push";
        cout<<"\n2:-pop";
        cout<<"\n3:-display";
        cout<<"\n4:exit";
        cout<<"\nenter choice";
        cin>>ch;
        switch(ch)
        {
                case 1: int no,flag=0;
                        cout<<"enter element" ;
                        cin>>no;
                        for(int i=0;i<5;i++)
                        {
                                if(arr[i]==0)
                                {
                                        arr[i]=no;
                                        flag=1;
                                        break;
                                }

                        }
                        if(flag==0)
```

```
                cout<<"stack is full";
                break;

        case 2: flag=0;
                for(i=4;i>=0;i--)
                {
                        if(arr[i]!=0)
                        {       cout<<arr[i];
                                arr[i]=0;
                                flag=1;
                                break;
                        }

                }
                if(flag==0)
                cout<<"stack is empty";
                break;

        case 3:for(i=0;i<5;i++)
               cout<<arr[i];
               break;
    //  case 4:exit(0);

    }
    }while(ch!=4);
}
```

Lab 2:-IMPLEMENTATION OF QUICK SORT

```
/* CJ9PR6.C: Quick sort. */


#include <stdio.h>
#include <conio.h>
```

```
int split ( int*, int, int ) ;

void main( )
{
    int arr[10] = { 11, 2, 9, 13, 57, 25, 17, 1, 90, 3 } ;
    int i ;

    void quicksort ( int *, int, int ) ;

    clrscr( ) ;

    printf ( "Quick sort.\n" ) ;
    printf ( "\nArray before sorting:\n") ;

    for ( i = 0 ; i <= 9 ; i++ )
            printf ( "%d\t", arr[i] ) ;

    quicksort ( arr, 0, 9 ) ;

    printf ( "\nArray after sorting:\n") ;

    for ( i = 0 ; i <= 9 ; i++ )
            printf ( "%d\t", arr[i] ) ;

    getch( ) ;
}

void quicksort ( int a[ ], int lower, int upper )
{
    int i ;
    if ( upper > lower )
```

```
    {
        i = split ( a, lower, upper ) ;
        quicksort ( a, lower, i - 1 ) ;
        quicksort ( a, i + 1, upper ) ;
    }
}

int split ( int a[ ], int lower, int upper )
{
    int i, p, q, t ;

    p = lower + 1 ;
    q = upper ;
    i = a[lower] ;

    while ( q >= p )
    {
        while ( a[p] < i )
            p++ ;

        while ( a[q] > i )
            q-- ;

        if ( q > p )
        {
            t = a[p] ;
            a[p] = a[q] ;
            a[q] = t ;
        }
    }
```

```
    t = a[lower] ;
    a[lower] = a[q] ;
    a[q] = t ;

    return q ;
}
```

Lab 3:-IMLPEMENTATION OF MIN AND MAX USING DIVIDE AND CONQUER METHOD.

```
#include<stdio.h>
int max, min;
int a[100];
```

```
void maxmin(int i, int j)


  {
    int max1, min1, mid;
    if(i==j)


            {
                    max = min = a[i];
            }
            else


            {
                    if(i == j-1)


                    {
                            if(a[i] <a[j])


                                    {
                                            max = a[j];
                                            min = a[i];
                                    }
                                    else


                                    {
                                            max = a[i];
```

```
                min = a[j];
            }
        }
        else


        {
            mid = (i+j)/2;
            maxmin(i, mid);
            max1 = max;   min1 = min;
            maxmin(mid+1, j);
            if(max <max1)
                max = max1;
            if(min > min1)
                min = min1;
        }
    }
}
void main ()


   {
    int i, num;
    clrscr();
    printf ("\n\t\t\tMAXIMUM & MINIMUM\n\n");
    printf ("\nEnter the total number of numbers : ");
    scanf ("%d",&num);
    printf ("Enter the numbers : \n");
    for (i=1;i<=num;i++)
```

```
        {
                scanf ("%d",&a[i]);
        }

        max = a[0];
        min = a[0];
        maxmin(1, num);
        printf ("Maximum element in an array : %d\n", max);
        printf ("Minimum element in an array : %d\n", min);
        getch();
}
```

LAB 4:-IMPLEMENTATION OF KNAPSACK PROBLEM USING GREEDY METHOD.

```
//knapsack problem
#include<iostream.h>
```

```
#include<conio.h>
#define m 20
void main()
{
    float n,i,a[100],b[100],weight[4],profit[4],sum,j,max;
    float fractions[4][3]={
                            0.5,0.33,0.25,
                            1,0.13,0,
                            0,0.66,1,
                            0,1,0.5
                            };

                           /* 0.5,0.33,0.25,
                            1,0.13,0,
                            0,0.61,1,
                            0,1,0.5
                           };    */
    cout<<"\t\t"<<"KNAPSACK PROBLEM";
    cout<<"\nenter how many objects you want in knapsack";
    cin>>n;
    if(n<m)
    {
            cout<<"\nenter"<<n<<"objects now";
            for(i=0;i<n;i++)
            cin>>a[i];
            cout<<"\nenter"<<n<<"weights now";
            for(i=0;i<n;i++)
            cin>>b[i];

            //steps to calculate weights
            for(i=0;i<4;i++)
```

```
{       sum=0;
          for(j=0;j<3;j++)
          {
                  sum=sum+b[j]*fractions[i][j];
          }
          weight[i]=sum;

}
//steps to calculate profits
for(i=0;i<4;i++)
{       sum=0;
          for(j=0;j<3;j++)
          {
                  sum=sum+a[j]*fractions[i][j];
          }
          profit[i]=sum;

}
max=profit[0];
for(i=0;i<4;i++)
{
 if(max<profit[i])
 max=profit[i];

}
cout<<"feasible solution is:->"<<max<<"\n";
for(i=0;i<4;i++)
cout<<"\n"<<weight[i];

}
else
```

```
    cout<<"max. no of objects possible are:->"<<m;

}
```

Lab 5:-IMPLEMENTATION OF MIN COST SPANNING TREE

```
#include<iostream.h>

#define MAX 10
```

```
class prims
{
    private : int cost[MAX][MAX], tree[MAX][MAX];
              int n;
    public  : void readmatrix();
              int spanningtree(int);
              void display(int);
};

void prims :: readmatrix()
{
    int i, j;
    cout << "\nEnter the number of vertices in the Graph : ";
    cin  >> n;
    cout << "\nEnter the Cost matrix of the Graph\n\n";
    for (i = 1; i <= n; i++)
            for (j = 1; j <= n; j++)
                    cin >> cost[i][j];
}

int prims :: spanningtree(int src)
{
    int visited[MAX], d[MAX], parent[MAX];
    int i, j, k, min, u, v, stcost;
    for (i = 1; i <= n; i++)
    {
            d[i] = cost[src][i];
            visited[i] = 0;
            parent[i] = src;
    }
    visited[src] = 1;
```

```
    stcost = 0;
    k = 1;
    for (i = 1; i < n; i++)
    {
            min = 999;
            for (j = 1; j <= n; j++)
            {
                    if (!visited[j] && d[j] < min)
                    {
                            min = d[j];
                            u = j;
                    }
            }
            visited[u] = 1;
            stcost = stcost + d[u];
            tree[k][1] = parent[u];
            tree[k++][2] = u;
            for (v = 1; v <= n; v++)
                    if (!visited[v] && (cost[u][v] < d[v]))
                    {
                            d[v] = cost[u][v];
                            parent[v] = u;
                    }
    }
    return (stcost);
}

void prims :: display(int cost)
{
    int i;
    cout << "\nThe Edges of the Mininum Spanning Tree are\n\n";
```

```
    for (i = 1; i < n; i++)
            cout << tree[i][1] << " " << tree[i][2] << endl;
    cout << "\nThe Total cost of the Minimum Spanning Tree is : " << cost;
}

int main()
{
    int source, treecost;
    prims pri;
    pri.readmatrix();
    cout << "\nEnter the Source : ";
    cin  >> source;
    treecost = pri.spanningtree(source);
    pri.display(treecost);
    return 0;
}
```

Lab 6:-IMPLEMENTATION OF SHORTEST PATH

```
/* CH10PR4.C: Program to find the shortes path. */

#include <stdio.h>
#include <conio.h>
```

```
#define INF 9999

void main( )
{
    int arr[4][4] ;
    int cost[4][4] = {
                                        7, 5, 0, 0,
                                        7, 0, 0, 2,
                                        0, 3, 0, 0,
                                        4, 0, 1, 0
                                    } ;
    int i, j, k, n = 4 ;

    clrscr( ) ;

    for ( i = 0 ; i < n ; i++ )
    {
        for ( j = 0; j < n ; j++ )
        {
            if ( cost[i][j] == 0 )
                arr[i][j] = INF ;
            else
                arr[i][j] = cost[i][j] ;
        }
    }

    printf ( "Adjacency matrix of cost of edges:\n" ) ;
    for ( i = 0 ; i < n ; i++ )
    {
        for ( j = 0; j < n ; j++ )
            printf ( "%d\t", arr[i][j] ) ;
```

```
            printf ( "\n" ) ;
    }

    for ( k = 0 ; k < n ; k++ )
    {
            for ( i = 0 ; i < n ; i++ )
            {
                    for ( j = 0 ; j < n ; j++ )
                    {
                            if ( arr[i][j] > arr[i][k] + arr[k][j] )
                                    arr[i][j] = arr[i][k] + arr[k][j];
                    }
            }
    }

    printf ( "\nAdjacency matrix of lowest cost between the vertices:\n" ) ;
    for ( i = 0 ; i < n ; i++ )
    {
            for ( j = 0; j < n ; j++ )
                    printf ( "%d\t", arr[i][j] ) ;
            printf ( "\n" ) ;
    }

    getch( ) ;
}
```

Lab 7:-IMPLEMENTATION OF MERGE SORT

```
public class MergeSorter
{
   private static int[] a, b;    // auxiliary array b
```

```
    public static void sort(int[] a0)
    {
        a=a0;
        int n=a.length;
        // according to variant either/or:
        b=new int[n];    b=new int[(n+1)/2];
        mergesort(0, n-1);
    }

    private static void mergesort(int lo, int hi)
    {
        if (lo<hi)
        {
            int m=(lo+hi)/2;
            mergesort(lo, m);
            mergesort(m+1, hi);
            merge(lo, m, hi);
        }
    }

    private static void merge(int lo, int m, int hi)
    {
    }

}    // end class MergeSorter
```

Lab 8:- IMLPEMENTATION OF 0/1 KNAPSACK PROBLEM

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int o[6],cost[6],t,x,sum=0,z,r,profit[6],p[6],y,p1,p2;
void main()
```

```
{
   int i,j,r,s=1,m;
   clrscr();
   cout<<"enter 5 objects";
   for(i=1;i<=5;i++)
     cin>>o[i];
   cout<<"enter cost for 5 objects";
   for(i=1;i<=5;i++)
   cin>>cost[i];
   cout<<"enter profit for 5 objects";
   for(i=1;i<=5;i++)
   cin>>p[i];
   cout<<"enter requirement";
   cin>>r;
   cout<<"solutions are:-";
   for(int k=1;k<=5;k++)
   {  s=1;
   for(i=k;i<=5;i++)
   {
    sum=0; j=k;m=0;
    for(int t=1;t<=s;t++)
    {
          sum=sum+cost[j];
          j++; m=j;
    }
    if(sum==r)
    {   ++y;
              cout<<"\n";
              for(j=k;j<m;j++)

              {
```

```
        if(y==1)
//      for(j=k;j<m;j++)
        cout<<o[j];

        profit[y]=profit[y]+p[j];
        }

         x=profit[1];
        int flag=0,ss=0;
          for(int hh=2;hh<y;hh++)
          {
            if(x<profit[hh])
            {
              x=profit[hh];
            }
          }
          if(x<profit[y])
          flag=1;
        if(flag==1)
        {
            for(j=k;j<m;j++)
            cout<<o[j];
          //   p1=k;p2=m;
        }




 //   }
//      exit(0);
```

```
      }
      s++;
     }

   }
   int max=profit[1];
   for(int h=2;h<=y;h++)
   {
     if(max<profit[h])
     max=profit[h];
   }
//   cout<<"\nfinal solution:->";
//   for(h=p1;h<p2;h++)
//   cout<<o[h];
   cout<<"\n"<<"Total Profit:->"<<max;
    getch();
}
```

Lab 9:-IMPLEMENTATION OF ALL PAIRS SHORTEST PATH

```
// Floyd's All pairs shortest path algorithm (O (n^3) )
// input is adjacency matrix output is matrix of shortest paths
// C is the adjacency matrix
// n is the order of the square matrix
```

```
// A is the all pairs shortest paths matrix
// we assume that A is allocated by the caller
int GraphAlgo::ComputeFloydAPSP(int *C, int n, int *A)
{

 int i,j,k;

 // set all connected positions to 1
 // and all unconnected positions to infinity
 for (i=0; i<n; i++)
 {
  for (j=0; j<n; j++)
  {
   if ( *(C+i*n+j) == 0)
   {
    *(A+i*n+j) = 999999999; // does this look like infinity ?
   }
   else
   {
    *(A+i*n+j) = 1;
   }
  }
 }

 // set the diagonals to zero
 for (i=0; i<n; i++)
 {
  *(A+i*n+i) = 0;
 }

 // for each route via k from i to j pick
```

```
  // any better routes and replace A[i][j]
  // path with sum of paths i-k and j-k
  for (k=0; k<n; k++)
  {
    for (i=0; i<n; i++)
    {
    for (j=0; j<n; j++)
    {
      if ( *(A+i*n+k) + *(A+k*n+j) < *(A+i*n+j) )
        {
          // A[i][j] = A[i][k] + A[k][j];
                    *(A+i*n+j) = *(A+i*n+k)+ *(A+k*n+j);
        }
      }
    }
  }

  return 0;
} // Floyd's algorithm


// this is for testing Floyd's algorithm
// demonstrates the allocation and
// deallocation of memory for the matrices
void FloydTest()
{

  // allocate the entire matrix in one linear array
  // trying to allocate it as an array of pointers to arrays
  // didnt quite work, possibly because the [] and * notation
  // arent quite replaceable
```

```
 int n=4;
 int *C=new int[n*n];
   C[0]=0; C[1]=1; C[2]=0; C[3]=0;
   C[4]=1; C[5]=0; C[6]=1, C[7]=0;
   C[8]=0; C[9]=1; C[10]=0;C[11]=1;
   C[12]=0;C[13]=0;C[14]=1;C[15]=0;

 int* A = new int[n*n];

 ComputeFloydAPSP (C,n,A);

 printf("Final shortest distances\n");
 for(int i=0;i<n;i++)
 {
   for(int j=0;j<n;j++)
   {
     printf("%d ",*(A+i*n+j));
   }
   printf("\n");
 }
 printf("End of All pairs Shortest paths\n");
   delete [] A;
   delete [] C;
```

Lab 10:- IMPLEMENTATION OF 8 QUEENS PROBLEM

```
# include<iostream.h>
#include<stdio.h>
#include<conio.h>
# define Q  8
int chkAll(int);
```

```
struct POINT {int x,y;}q[Q];
//POINT q[Q];
int N;
int chkAll(int x)
{
    for ( int i =x ; i >=0 ; i--)

    for(int j = i - 1; j >= 0 ; j--)
    if(q[i].x == q[j].x || q[i].y == q[j].y ||
      q[i].x + q[i].y == q[j].x + q[j].y ||
      q[i].x - q[j].x == q[i].y - q[j].y)
    return 0;
    return 1;
}
void MoveQueen(int x)
{

    if(x >= Q)
    {
            printf("\n\nSolution : %d \n\n\t",++N);

            for(int j=0;j<Q;printf("\n\t"),j++)
            for(int i=0;i<Q;((q[j].x==j) && (q[j].y==i))?printf("Q%c",179) :
printf("_%c",179),i++);
            return;
    }
    int s;
    for (int j = 0; j < Q ;q[x].x = x,j++){   q[x].y = j;
             if((s=chkAll(x))==1) MoveQueen(x + 1);
    }
}
```

```
void main()
{

for (int i = 0; i < Q; q[0].x = 0,q[0].y = i,MoveQueen(1),i++);
clrscr();

}
```

4. Quiz on the subject:

Quiz should be conducted on tips in the laboratory, recent trends and subject knowledge of the subject. The quiz questions should be formulated such that questions are normally are from the scope outside of the books. However twisted questions and self formulated questions by the faculty can be asked but correctness of it is necessarily to be thoroughly checked before the conduction of the quiz.

5. Conduction of Viva-Voce Examinations:

Teacher should oral exams of the students with full preparation. Normally, the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested Oral examinations are to be conducted in co-cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be

offered to each other in case of difference of opinion, which should be critically suppressed in front of the students.

6. Submission:

Document Standard:

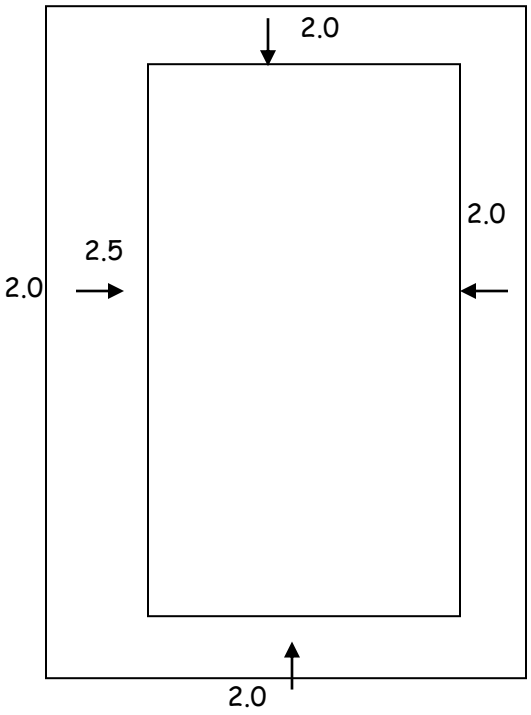
A] Page Size — A4 Size

B] Running text — Justified text

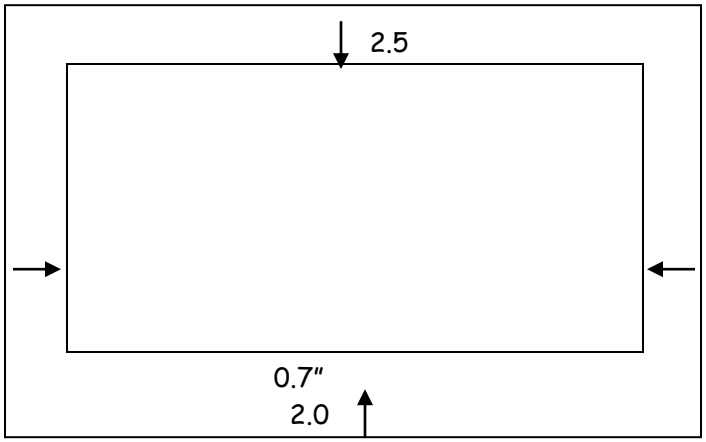
C] Spacing — 1 Line

D] Page Layout and Margins (Dimensions in Cms)

Normal Page



Horizantal



Desription	Font	Size	Boldness	Italics	Underline	Capitalize
College Name	Arial	24	-----	------	Yes	----------
Document Title	Tahoma	22	-----	------	---------	----------
Document Subject	Century Gothic	14	-----	------	---------	Capital
Class	Bookman old Slyle	12	-----	------	---------	----------
Document No	Bookman old Slyle	10	-----	------	---------	----------
Copy write inf	Bookman old Slyle	9	-----	------	---------	----------
Forward heading	Bookman old Slyle	12	-----	------	Yes	Capital
Forward matter	Bookman old Slyle	12	-----	------	---------	----------
Lab man Contents title	Bookman old Slyle	12	-----	------	Yes	Capital
Index title	Bookman old Slyle	12	Yes	------	Yes	Capital

Index contents	Bookman old Slyle	12	-----	------	---------	----------
Heading	Tahoma	14	Yes	Yes	Yes	----------
Running Matter	Comic Sans MS	10	-----	------	---------	----------

7. Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become popular amongst the students. It is a wrong approach or concept to award the students by way of easy marking to get cheap popularity among the students to which they do not deserve. It is a primary responsibility of the teacher that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with unjust circumstances.