1. **Simulate a railway time table at an intermediate station with N trains, M destinations and K routes. Each train has a fixed arrival and departure time at the station which may change due to certain reasons causing a delay in arrival (and hence departure). The delay of any train causing overlapping in timing must be dealt with giving priority to some train (to be decided by the programmer). No train is supposed to reach before its arrival time at the station. Apply elementary data structures to store and update the time table which is displayed on the display board continuously with updates after every 5 minutes. If a train departs, its name is eliminated from the display and if a new train comes within the frame of 1 hour (say), its entry is done on the display. You may assume the frame of display with other requirements for the simulation of such a time table.**

   Explanation - For the above problem, i created the Structure of train which contains 6 variable id, arr_hr, arr_min, dep_hr, dep_min, delay and then declared the array of these structure which are basically the trains instance and then added the delay min in both arrival time and departure time after that I sorted the Trains according the arrival time I handled time overlapping condition simply by putting,

```
For ( All trains except the first train )
{
If (arrival time of new train < departure time of previous train)
Then
{
arrival time of new train = arrival time of new train + 1min + (departure time of previous train subtracted by arrival time of new train)
departure time of new train = departure time of new train + 1min + (departure time of previous train subtracted by arrival time of new train)
}
}
```

   then, I printed the updated status of all trains .

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

struct Train
{
   int id;
   int arr_hr;
   int arr_min;
   int dep_hr;
```

```
    int dep_min;
    int delay;
};

using namespace std;

void swap(Train *A, Train *B){
    Train *temp;
    *temp = *A;
    *A = *B;
    *B = *temp;
}


bool compare_two_trains(Train a, Train b)
{

    if (a.arr_hr < b.arr_hr )
        return 1;


    if (a.arr_hr > b.arr_hr  )
        return 0;


    if (a.arr_hr == b.arr_hr )
    if (a.arr_min > b.arr_min )
        return 0;

}
/* Data used

Train_id      arrival_hr:arrival_min      departure_hr:departurel_min     delay_min
0                       10:00                          10:10                 20
1                       10:10                          10:20                 15
2                       10:30                          10:40                 10
3                       10:35                          10:45                 10
4                       11:20                          11:40                 15



*/
```

```
int main()
{
   int n=5;

   Train all[5];
   all[0].id = 0; all[0].arr_hr = 10; all[0].arr_min = 00; all[0].dep_hr = 10; all[0].dep_min = 10;
all[0].delay = 20;
   all[1].id = 1; all[1].arr_hr = 10; all[1].arr_min = 10; all[1].dep_hr = 10; all[1].dep_min = 20;
all[1].delay = 15;
   all[2].id = 2; all[2].arr_hr = 10; all[2].arr_min = 30; all[2].dep_hr = 10; all[2].dep_min = 40;
all[2].delay = 10;
   all[3].id = 3; all[3].arr_hr = 10; all[3].arr_min = 35; all[3].dep_hr = 10; all[3].dep_min = 45;
all[3].delay = 10;
   all[4].id = 4; all[4].arr_hr = 11; all[4].arr_min = 20; all[4].dep_hr = 11; all[4].dep_min = 40;
all[4].delay = 15;

   std::cout << "Time table of trains at Allahabad Station and current time is 09:59:59"<<
std::endl;
   std::cout << "Train_id        arrival_hr:arrival_min        departure_hr:departurel_min
delay_min"<< std::endl;
   std::cout << "0                 10:00                    10:10                 20"<< std::endl;
   std::cout << "1                 10:10                    10:20                 15"<< std::endl;
   std::cout << "2                 10:30                    10:40                 10"<< std::endl;
   std::cout << "3                 10:35                    10:45                 10"<< std::endl;
   std::cout << "4                 11:20                    11:40                 15"<< std::endl;


   int curr_hr=10;
   int curr_min=00;

   for(int i=0; i<n; i++){
      if(all[i].arr_min+all[i].delay < 60)
      {
      all[i].arr_min=all[i].arr_min+all[i].delay%60;
      }
      else
      {
      all[i].arr_min=all[i].arr_min+all[i].delay%60;
      all[i].arr_hr=(all[i].arr_hr + 1) % 24;
      }

      if(all[i].dep_min+all[i].delay < 60)
```

```
        {
        all[i].dep_min=all[i].dep_min+all[i].delay%60;
        }
        else
        {
        all[i].dep_min=all[i].dep_min+all[i].delay%60;
        all[i].dep_hr=(all[i].dep_hr + 1) % 24;
        }



    }

     sort(all, all+5, compare_two_trains);

   for(int i=1 ; i<n ;i++){
    if(all[i-1].dep_hr == all[i].arr_hr)
    if(all[i-1].dep_min > all[i].arr_min)
    all[i].arr_min+=all[i-1].dep_min - all[i].arr_min+1;
    all[i].dep_min+=all[i-1].dep_min - all[i].arr_min+1;
    }

cout<<"===========================================================================
======="<<endl;
        //Printing Station Status;
    std::cout << "Train_id          arrival_hr:arrival_min     departure_hr:departurel_min"<<
std::endl;
          for(int i=0; i<n; i++)
     {
    cout<<all[i].id <<"                     "<<all[i].arr_hr<<":"<<all[i].arr_min<<"
"<<all[i].dep_hr<<":"<<all[i].dep_min<<endl;
     }
    return 0;
}
```

2. Usually a compiler is restricted to process 16-bit numbers (producing 10-digit decimal numbers) for integer arithmetic. For certain calculations, very large numbers are required, e.g., 20-digit numbers or more. Suggest a scheme using elementary data structures to store such large numbers and perform the basic operations on the numbers. Estimate factorial of 31 (31!) using the scheme.

Explanation – The given C program can print any factorial which contain up to 20000 digits and every out is precise upto 20000 digits.

```
#include <stdio.h>


int mul_by_n_minus_one(int *num, int n_minus_one , int count_digits){

  int temp=0;

  int index=0;

  int y=0;


  for(int i = 0; i<count_digits; i++){

    y=num[index]*n_minus_one+temp;

    num[index]=y%10;

    temp=y/10;

    index++;

  }


  while(temp>0){

    num[count_digits]=temp%10;

    temp=temp/10;

    count_digits++;

  }
```

```
    return count_digits;



}

int digits_in_input(int n)
{
    int count_digits;
         while (n != 0)
    {
        n /= 10;
        ++count_digits;
    }
    printf("%d", count_digits);
    printf("\n");
    return count_digits;
}



int main()
{   int n;
    int num[20000];
    int num_size=1;
    int count_digits;
    num[0]=1;
```

```
int fac,temp;

for(int i=0;i<20000;i++)

{

   num[i]=10;

}


scanf(" %d", &fac);

count_digits=digits_in_input(fac);


if(fac==1 || fac==0)

{

   //printf("Factorial of given no. is: 1");

}


else if(fac>1)

{

   int i=0;

   int rem,value;

   value=fac;

   while(i<count_digits)

   {

      rem=value%10;

      num[i]=rem;

      value= value/10;

      i++;
```

```
    }

    int loop = fac - 1;

    int new_size;

    printf("Saved digits in array: ");

 for(int i=0;i<count_digits;i++)

 {

    printf("%d", num[i]);

 }

 printf("\n");


// printf("loop %d count_digits %d", loop,count_digits);

 //printf("\n");


 new_size=count_digits;


 while(loop>1){

    new_size = mul_by_n_minus_one(num,loop,new_size);

    loop--;

 }


 printf("No. of digits in output : %d", new_size);

 printf("\n");


 int print;

 print=new_size;
```

```
    int k=new_size - 1;

    for(int i=0;i<new_size;i++)

    {

        printf("%d", num[k]);

        k--;

    }


    }

    else

    {


    }


    return 0;

}
```