ABSTRACT

In this project we are going to implement Railway Reservation System. In this we have four modules: first module, we are going to take atleast 50 train numbers and sort them by using **Bubble Sort** algorithm and in the second module, we take at least 50 train names and sort them using **Selection Sort** algorithm and in the third module, we take at least 50 passenger names travelling in the train and sort their names using **Insertion Sort** algorithm and in the fourth module, we take at least 50 PNR numbers of the trains and sort them using **Quick Sort** algorithm. In all the four modules we take an array data structure to store all the required data in all the above four modules and construct a C function. Next we check the above functions by giving different inputs and fix the bugs if found.

INTRODUCTION

Sorting is, without doubt, the most fundamental algorithmic problem that was faced in the early days on computing. In fact, most of the computer science research was centered on finding a best way to sort a set of data. There is probably a good reason to make sorting that important.

1. Supposedly, 25% of all CPU cycles are spent sorting
2. Sorting is fundamental to most other algorithmic problems, for example binary search.
3. Many different approaches lead to useful sorting algorithms, and these ideas can be used to solve many other problems.

There are so many things in our real life that we need to search for, like a particular record in database, roll numbers in merit list, a particular telephone number in telephone directory, a particular page in a book etc. All this would have been a mess if the data was kept unordered and unsorted, but fortunately the concept of sorting came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search.

There are many applications of sorting. Once a list is sorted many questions about the list can be answered easily. We can efficiently find an element in a sorted list using Binary Search. Binary search requires only O(log n) operations in finding an element. We can also determine in O(n) if a sorted list has duplicates. We can construct a frequency distribution of the list if the list is sorted, or find the median and mode of the list in O(1) and O(n) respectively. We can find the k^{th} largest element in a list in O(1) time.

MODULE DESCRIPTION

In first module, we are going to take at least 50 train numbers and sort them by using **Bubble Sort** algorithm. Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list, compares adjacent pairs and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. $O(n^2)$ is the time complexity as there are two nested loops.

In the second module, we take at least 50 train names and sort them using **Selection Sort** algorithm. The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two sub arrays in a given array. The sub array which is already sorted. Remaining sub array which is unsorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub array is picked and moved to the sorted sub array. $O(n^2)$ is the time complexity as there are two nested loops.

In the third module, we take at least 50 passenger names travelling in the train and sort their names using **Insertion Sort** algorithm**.** Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands. Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as quick sort, heap sort, or merge sort. Efficient for (quite) small data sets, much like other quadratic sorting algorithms. O(n*2) is the time complexity as there will be 2 for loops.

In the fourth module, we take at least 50 PNR numbers of the trains and sort them using **Quick Sort** algorithm. Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quick Sort that pick pivot in different ways. Always pick first element as pivot, always pick last element as pivot (implemented below), pick a random element as pivot, pick median as pivot.The key process in quick Sort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

FUNCTIONAL REQUIREMENTS

HARDWARE

1. Processor – Intel Core duo

2. RAM – 1GB

3. Hard Disk – 100GB

SOFTWARE

1. Operating System – Windows 10

2. turbo c++

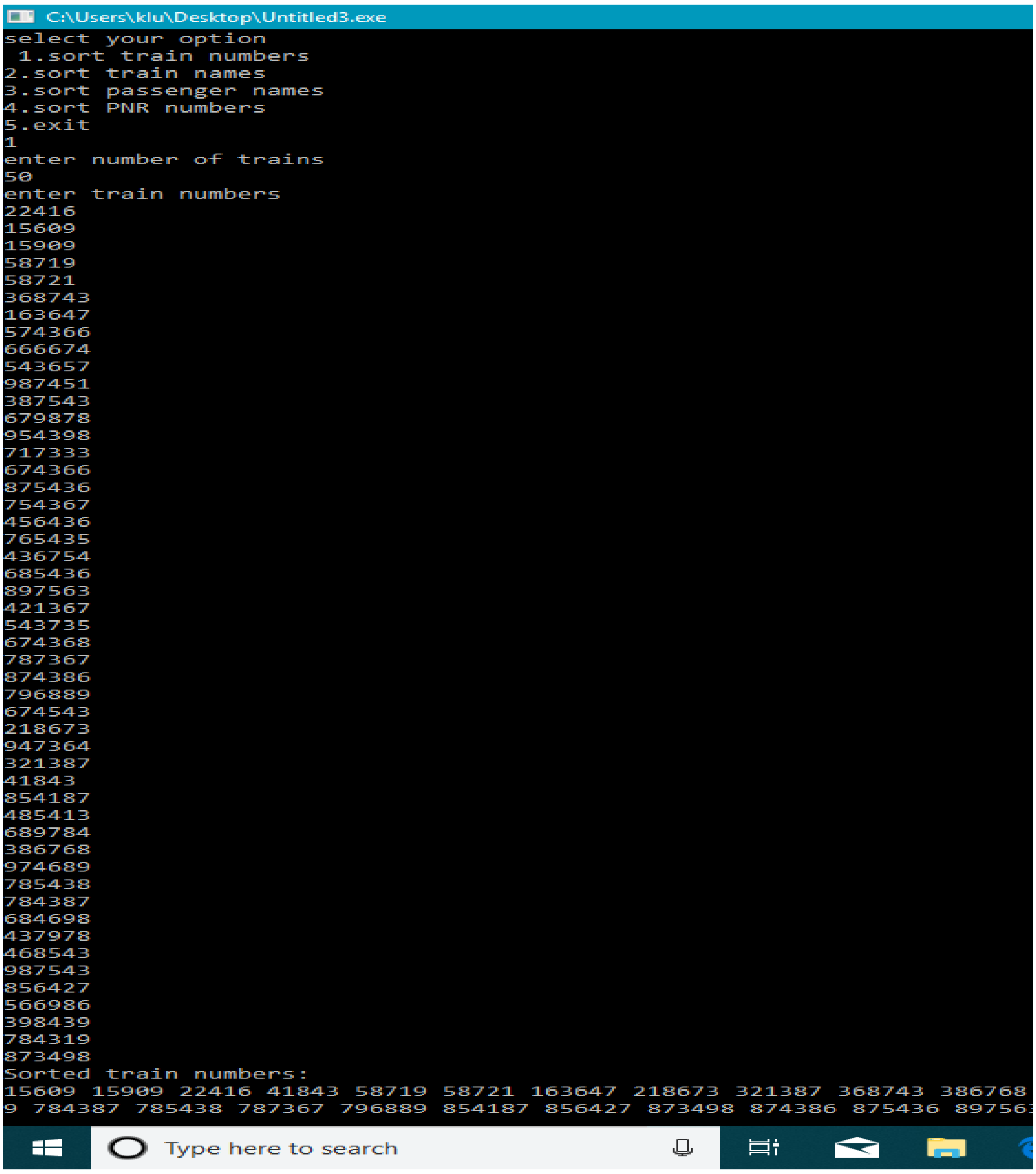
SOURCE CODE

```
#include <stdio.h>

#include<string.h>
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void Longswap(long int *xp, long int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("n");
}
int partition (long int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high- 1; j++)
    {

        if (arr[j] <= pivot)
        {
            i++;
            Longswap(&arr[i], &arr[j]);
        }
    }
    Longswap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void quickSort(long int arr[], int low, int high)
```

```
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
void printLongArray(long int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%ld ", arr[i]);
    printf("n");
}
int main()
{int n;
    printf("select your option\n 1.sort train numbers\n2.sort train names\n3.sort
passenger names\n4.sort PNR numbers\n5.exit\n");
    int ch;
    do{
       scanf("%d",&ch);
       if(ch==1)

          { printf("enter number of trains\n");
           scanf("%d",&n);
           printf("enter train numbers\n");
           int a[n];
           for(int i=0;i<n;i++)
           scanf("%d",&a[i]);
            bubbleSort(a, n);
            printf("Sorted train numbers: \n");
            printArray(a, n); }
      if(ch==2)

       { char name[100][100], tname[100][100], temp[100];
        int i, j, n;
        printf("Enter number of train names\n");
        scanf("%d", &n);
        printf("Enter %d names \n", n);
        for (i = 0; i < n; i++)
        {
            scanf("%s", name[i]);
            strcpy(tname[i], name[i]);
        }
        for (i = 0; i < n - 1 ; i++)
        {
            for (j = i + 1; j < n; j++)
            {
                if (strcmp(name[i], name[j]) > 0)
                {
```

```
                strcpy(temp, name[i]);
                strcpy(name[i], name[j]);
                strcpy(name[j], temp);
            }
        }
    }
    printf("\n----------------------------------------\n");
    printf("Input NamestSorted names\n");
    printf("-----------------------------------------\n");
    for (i = 0; i < n; i++)
    {
        printf("%s\t\t%s\n", tname[i], name[i]);
    }
    printf("-----------------------------------------\n");
   }
   if(ch==3)
   {
       char name[100][100], tname[100][100];
    int i, j, n;
    printf("Enter number of passengers \n");
    scanf("%d", &n);
    printf("Enter %d names \n", n);
    for (i = 0; i < n; i++)
    {
        scanf("%s", name[i]);

    }
    for (i = 1; i < n  ; i++)
    {
        j=i-1;
        strcpy(tname[i], name[i]);

            while (j>=0&&name[j]>tname[i])
        {
            strcpy(name[j+1], name[j]);
         j=j-1;
        }
        strcpy(name[j+1],tname[i]);

    }
    printf("\n----------------------------------------\n");
    printf("Input NamestSorted names\n");
    printf("-----------------------------------------\n");
    for (i = 0; i < n; i++)
    {
        printf("%s\t\t%s\n", tname[i], name[i]);
    }
    printf("-----------------------------------------\n");
  }
  if(ch==4)
  {
        int num;
        printf("enter number of PNR numbers to be sorted\n");
        scanf("%d",&num);
        long int a[n];
        printf("enter PNR numbers\n");
        for(int i=0;i<num;i++)
        scanf("%ld",&a[i]);
        quickSort(a, 0, num-1);
        printf("Sorted PNR numbers: \n");
        printLongArray(a, num);
    }


}while(ch<=5);
return 0;}
```

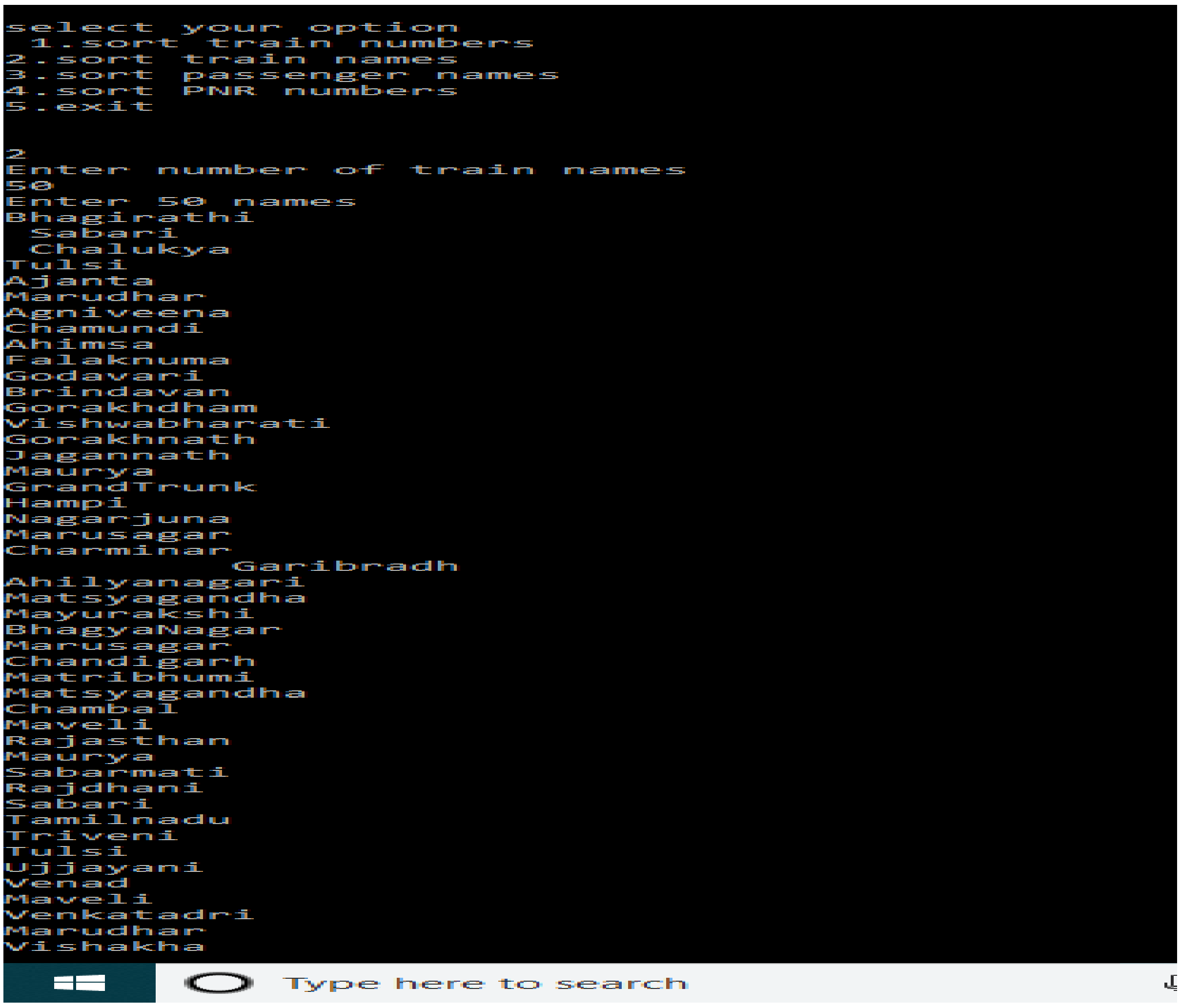
RESULT



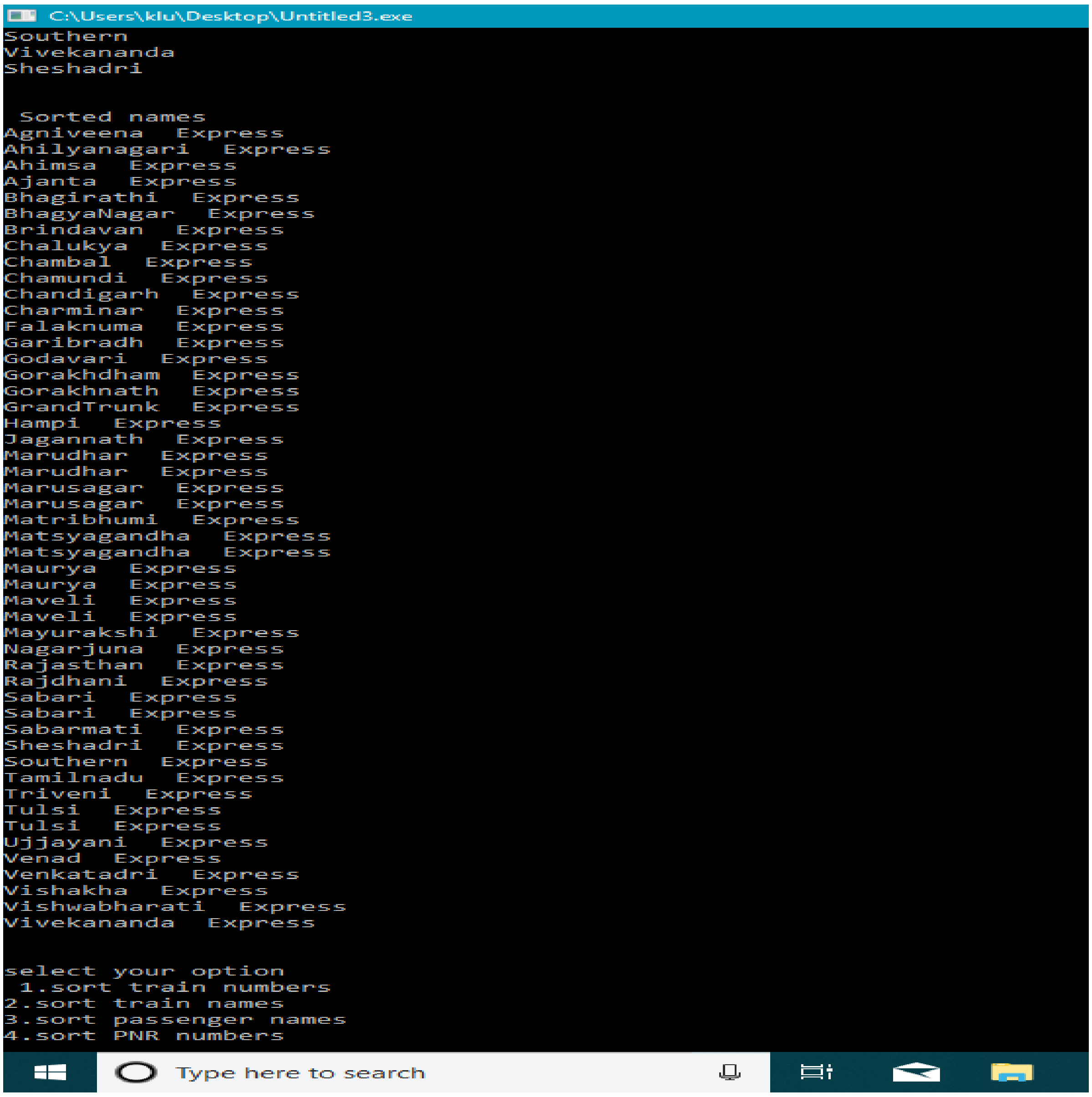
C:\Users\klu\Desktop\Untitled3.exe

```
select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit
1
enter number of trains
50
enter train numbers
22416
15609
15909
58719
58721
368743
163647
574366
666674
543657
987451
387543
679878
954398
717333
674366
875436
754367
456436
765435
436754
685436
897563
421367
543735
674368
787367
874386
796889
674543
218673
947364
321387
41843
854187
485413
689784
386768
974689
785438
784387
684698
437978
468543
987543
856427
566986
398439
784319
873498
Sorted train numbers:
15609 15909 22416 41843 58719 58721 163647 218673 321387 368743 386768
9 784387 785438 787367 796889 854187 856427 873498 874386 875436 89756
```

Type here to search

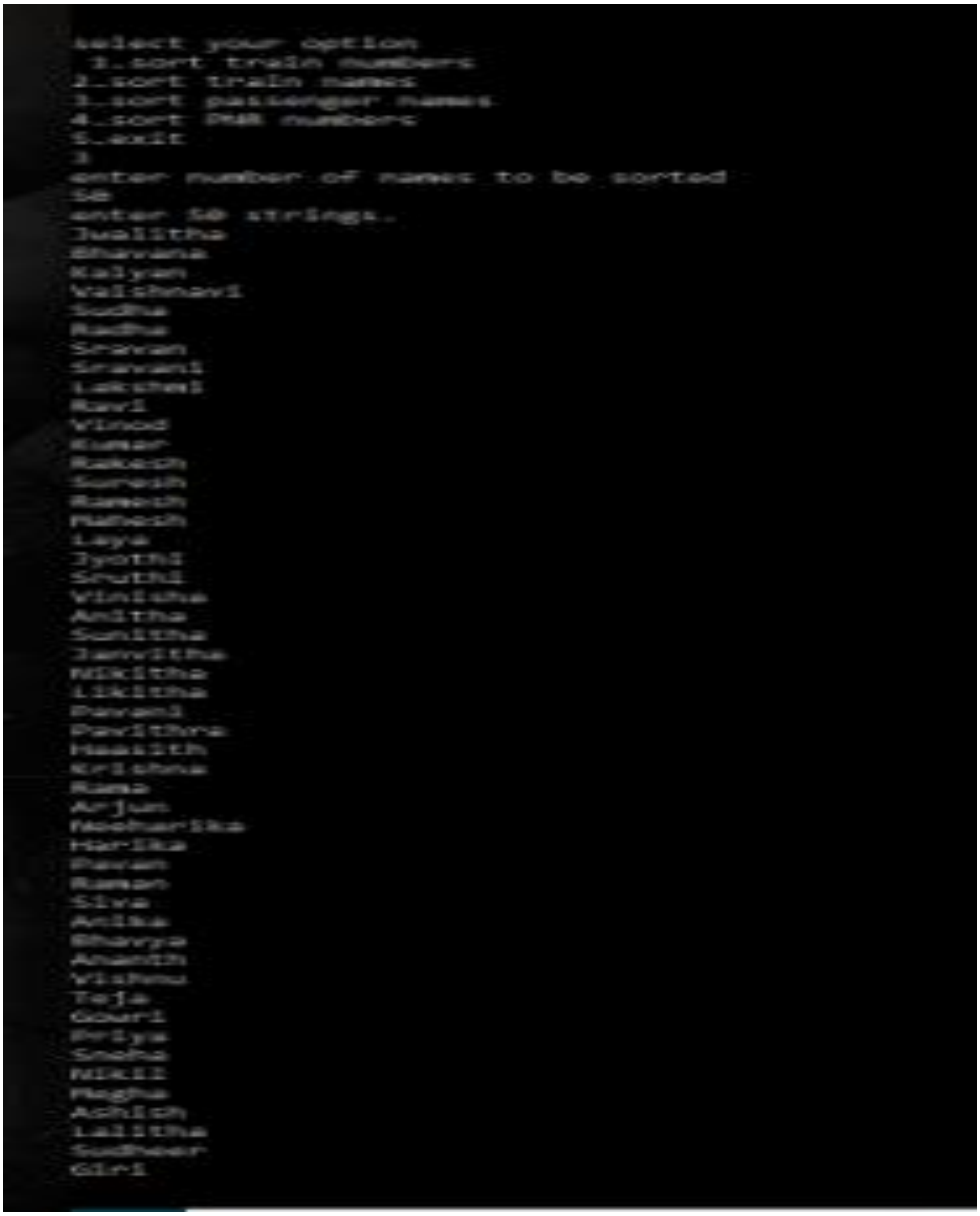


```
select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit


2
Enter number of train names
50
Enter 50 names
Bhagirathi
 Sabari
 Chalukya
Tulsi
Ajanta
Marudhar
Agniveena
Chamundi
Ahimsa
Falaknuma
Godavari
Brindavan
Gorakhdham
Vishwabharati
Gorakhnath
Jagannath
Maurya
GrandTrunk
Hampi
Nagarjuna
Marusagar
Charminar
                  Garibradh
Ahilyanagari
Matsyagandha
Mayurakshi
BhagyaNagar
Marusagar
Chandigarh
Matribhumi
Matsyagandha
Chambal
Maveli
Rajasthan
Maurya
Sabarmati
Rajdhani
Sabari
Tamilnadu
Triveni
Tulsi
Ujjayani
Venad
Maveli
Venkatadri
Marudhar
Vishakha
```

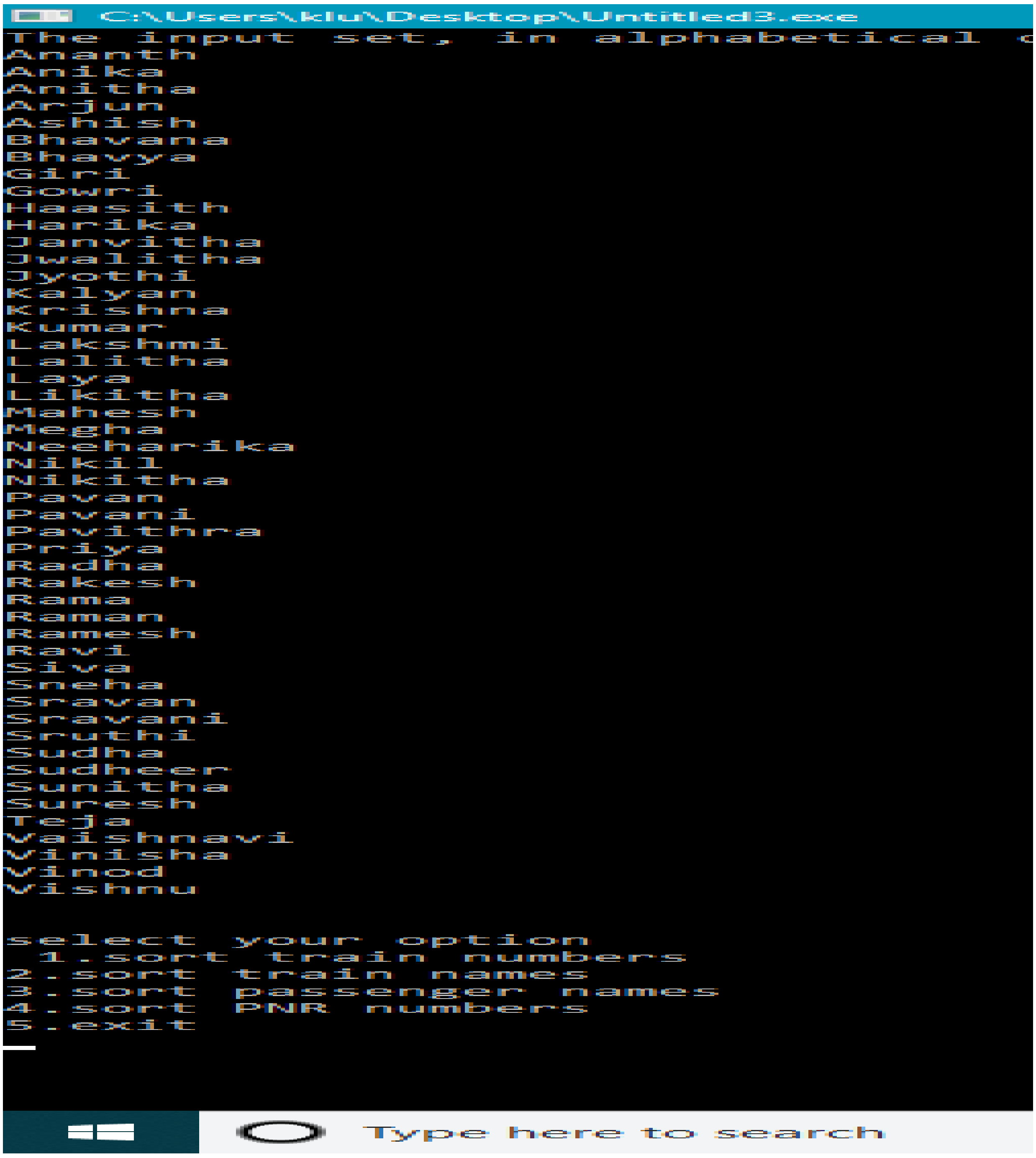


C:\Users\klu\Desktop\Untitled3.exe

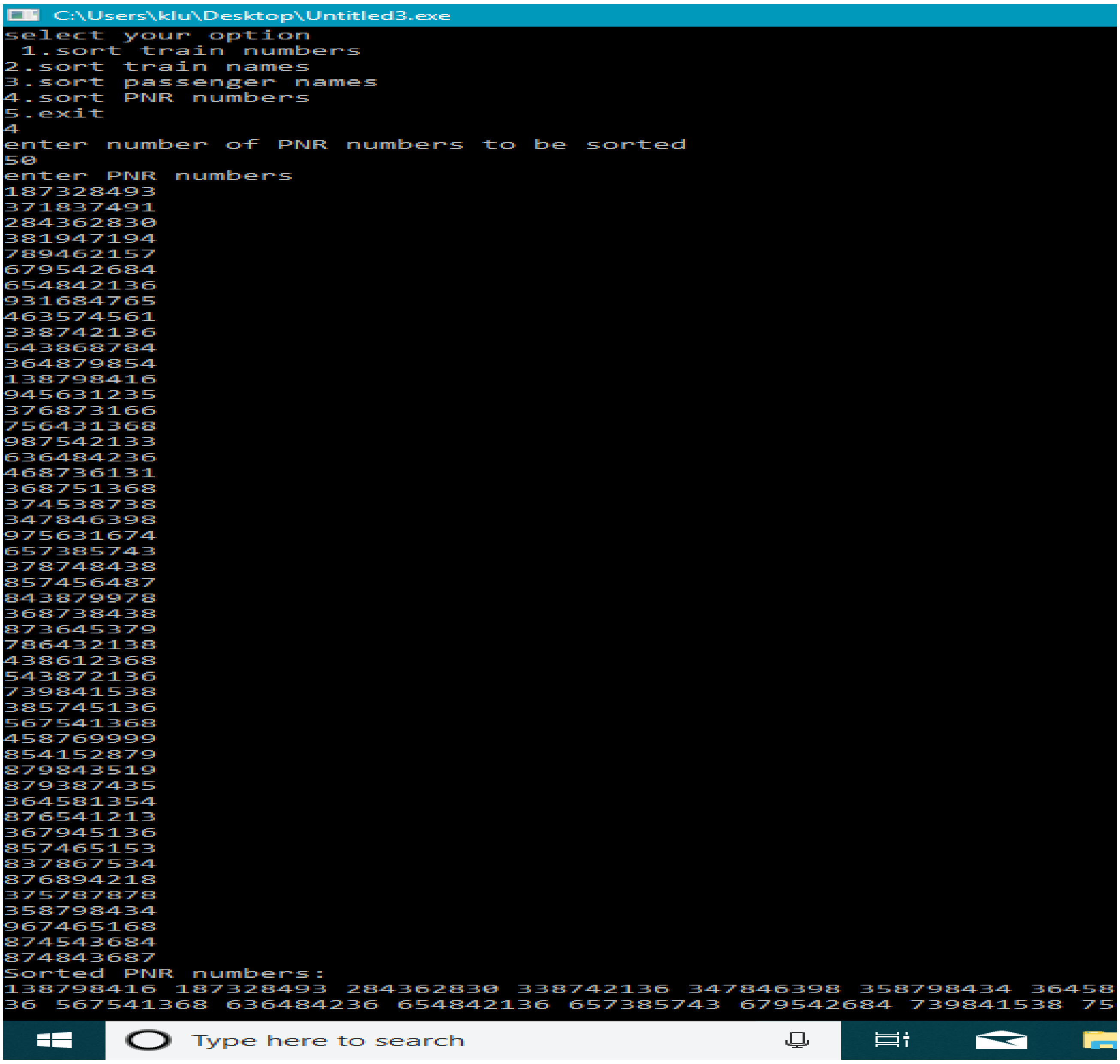
```
Southern
Vivekananda
Sheshadri


 Sorted names
Agniveena   Express
Ahilyanagari   Express
Ahimsa   Express
Ajanta   Express
Bhagirathi   Express
BhagyaNagar   Express
Brindavan   Express
Chalukya   Express
Chambal   Express
Chamundi   Express
Chandigarh   Express
Charminar   Express
Falaknuma   Express
Garibradh   Express
Godavari   Express
Gorakhdham   Express
Gorakhnath   Express
GrandTrunk   Express
Hampi   Express
Jagannath   Express
Marudhar   Express
Marudhar   Express
Marusagar   Express
Marusagar   Express
Matribhumi   Express
Matsyagandha   Express
Matsyagandha   Express
Maurya   Express
Maurya   Express
Maveli   Express
Maveli   Express
Mayurakshi   Express
Nagarjuna   Express
Rajasthan   Express
Rajdhani   Express
Sabari   Express
Sabari   Express
Sabarmati   Express
Sheshadri   Express
Southern   Express
Tamilnadu   Express
Triveni   Express
Tulsi   Express
Tulsi   Express
Ujjayani   Express
Venad   Express
Venkatadri   Express
Vishakha   Express
Vishwabharati   Express
Vivekananda   Express


select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
```



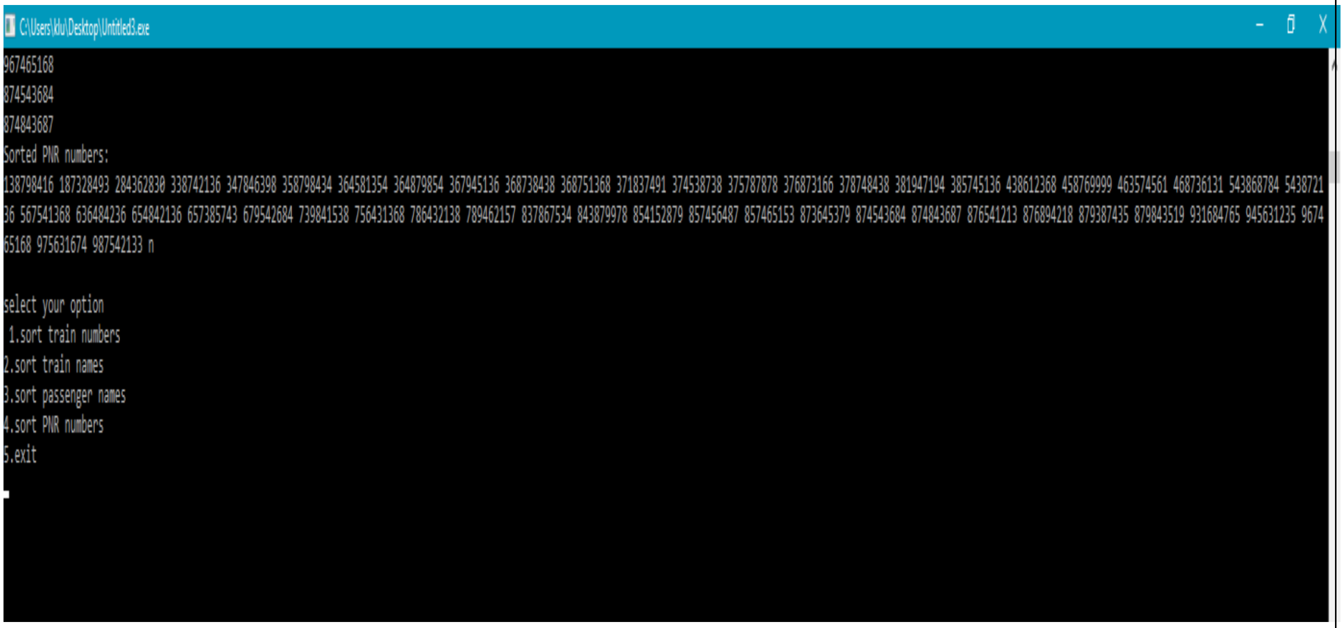
```
select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit
3
enter number of names to be sorted
50
enter 50 strings..
Jwalitha
Bhavana
Kalyan
Vaishnavi
Sudha
Radha
Sravan
Sravani
Lakshmi
Ravi
Vinod
Kumar
Rakesh
Suresh
Ramesh
Mahesh
Laya
Jyothi
Sruthi
Vinisha
Anitha
Sunitha
Janvitha
Nikitha
Likitha
Pavani
Pavithra
Hasith
Krishna
Rama
Arjun
Mosharika
Harika
Pavan
Raman
Siva
Anika
Bhavya
Ananth
Vishnu
Teja
Gowri
Priya
Sneha
Nikil
Megha
Ashish
Lalitha
Sudheer
Giri
```



```
The input set, in alphabetical o
Ananth
Anika
Anitha
Arjun
Ashish
Bhavana
Bhavya
Giri
Gowri
Haasith
Harika
Janvitha
Jwalitha
Jyothi
Kalyan
Krishna
Kumar
Lakshmi
Lalitha
Laya
Likitha
Mahesh
Megha
Neeharika
Nikil
Nikitha
Pavan
Pavani
Pavithra
Priya
Radha
Rakesh
Rama
Raman
Ramesh
Ravi
Siva
Sneha
Sravan
Sravani
Sruthi
Sudha
Sudheer
Sunitha
Suresh
Teja
Vaishnavi
Vinisha
Vinod
Vishnu

select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit
```



```
select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit
4
enter number of PNR numbers to be sorted
50
enter PNR numbers
187328493
371837491
284362830
381947194
789462157
679542684
654842136
931684765
463574561
338742136
543868784
364879854
138798416
945631235
376873166
756431368
987542133
636484236
468736131
368751368
374538738
347846398
975631674
657385743
378748438
857456487
843879978
368738438
873645379
786432138
438612368
543872136
739841538
385745136
567541368
458769999
854152879
879843519
879387435
364581354
876541213
367945136
857465153
837867534
876894218
375787878
358798434
967465168
874543684
874843687
Sorted PNR numbers:
138798416 187328493 284362830 338742136 347846398 358798434 36458
36 567541368 636484236 654842136 657385743 679542684 739841538 75
```



```
C:\Users\klu\Desktop\Untitled3.exe
967465168
874543684
874843687
Sorted PNR numbers:
138798416 187328493 284362830 338742136 347846398 358798434 364581354 364879854 367945136 368738438 368751368 371837491 374538738 375787878 376873166 378748438 381947194 385745136 438612368 458769999 463574561 468736131 543868784 5438721
36 567541368 636484236 654842136 657385743 679542684 739841538 756431368 786432138 789462157 837867534 843879978 854152879 857456487 857465153 873645379 874543684 874843687 876541213 876894218 879387435 879843519 931684765 945631235 9674
65168 975631674 987542133 n

select your option
 1.sort train numbers
2.sort train names
3.sort passenger names
4.sort PNR numbers
5.exit
```

CONCLUSION

By the end of the project result, sorting of the train names, train numbers, passenger names, PIN number by using various sorting technique bubble sort selection sort, insertion sort, quick sort. Sorting is important in programming for the same reason it is important in everyday life. It is easier and faster to locate items in a sorted list than unsorted. Sorting algorithms can be used in a program to sort an array for later searching or writing out to an ordered file or report. In future too, there will be companies and MNC's using the completely duration time and process utilization approach in every facility. This project based on the sorting method, in future there will be easier and advanced approach.

REFERENCES

www.wikipedia.org

www.Scribd.com

www.Handouts.org

www.google.com