DESIGN AND ANALYSIS OF ALGORITHMS – 2CS503

Practical 1

Name: Bhanderi Mihir

Roll No.: 19BCE023

Batch No.: A-1

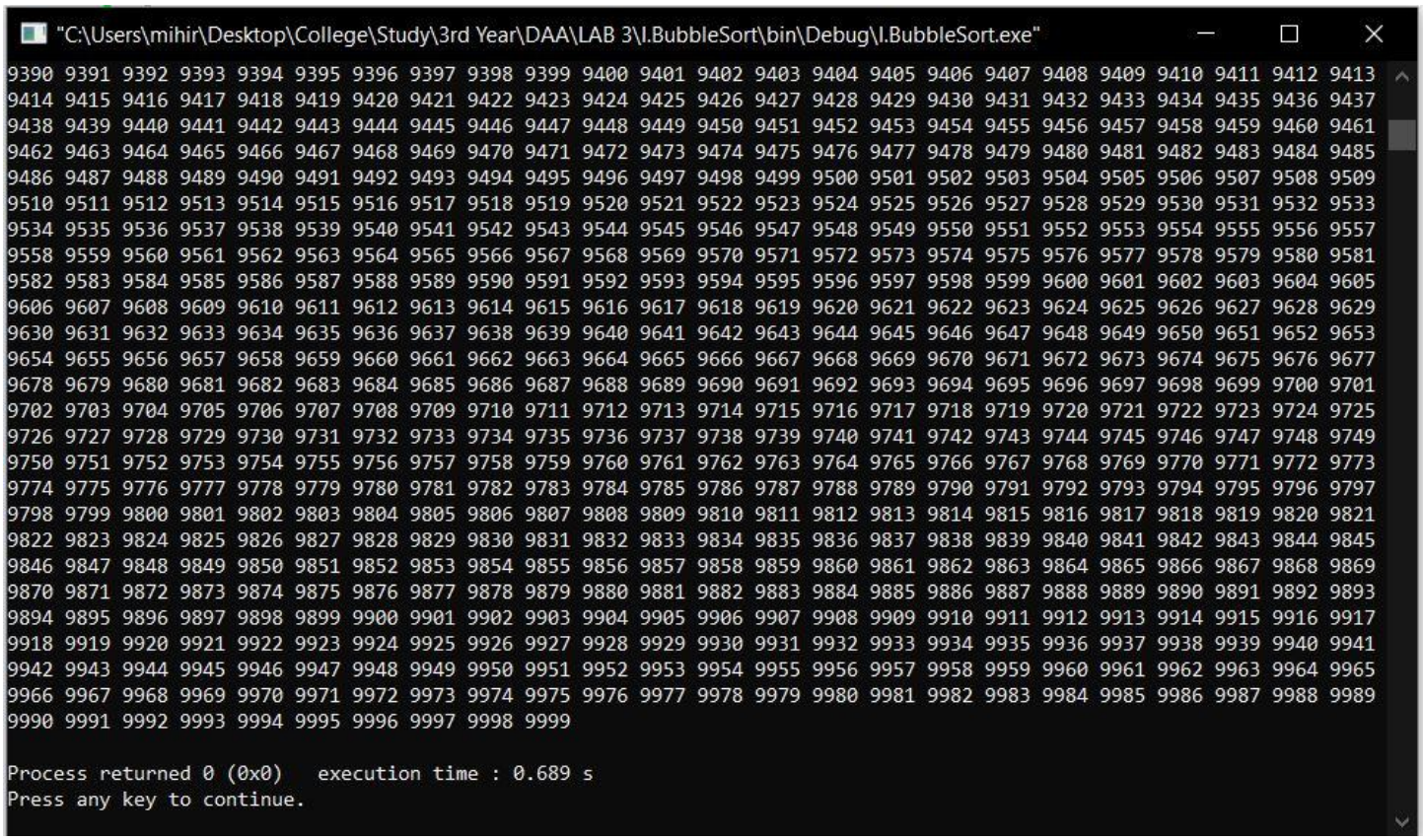
1. Iterative Bubble Sort

Code:

```
#include <stdio.h>
#include<stdlib.h>
#include<time.h>

void swap(int xp, 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++)
```

```
{
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 main()
{
int no=10000;
int j = no;
int arr[no];
for(int i=0;i<no;i++)
{
arr[i]=no-j;

j--;
}
```

```
int n = sizeof(arr)/sizeof(arr[0]);
bubbleSort(arr, n);
printf("Sorted array: \n");
printArray(arr, n);

return 0;
}
```



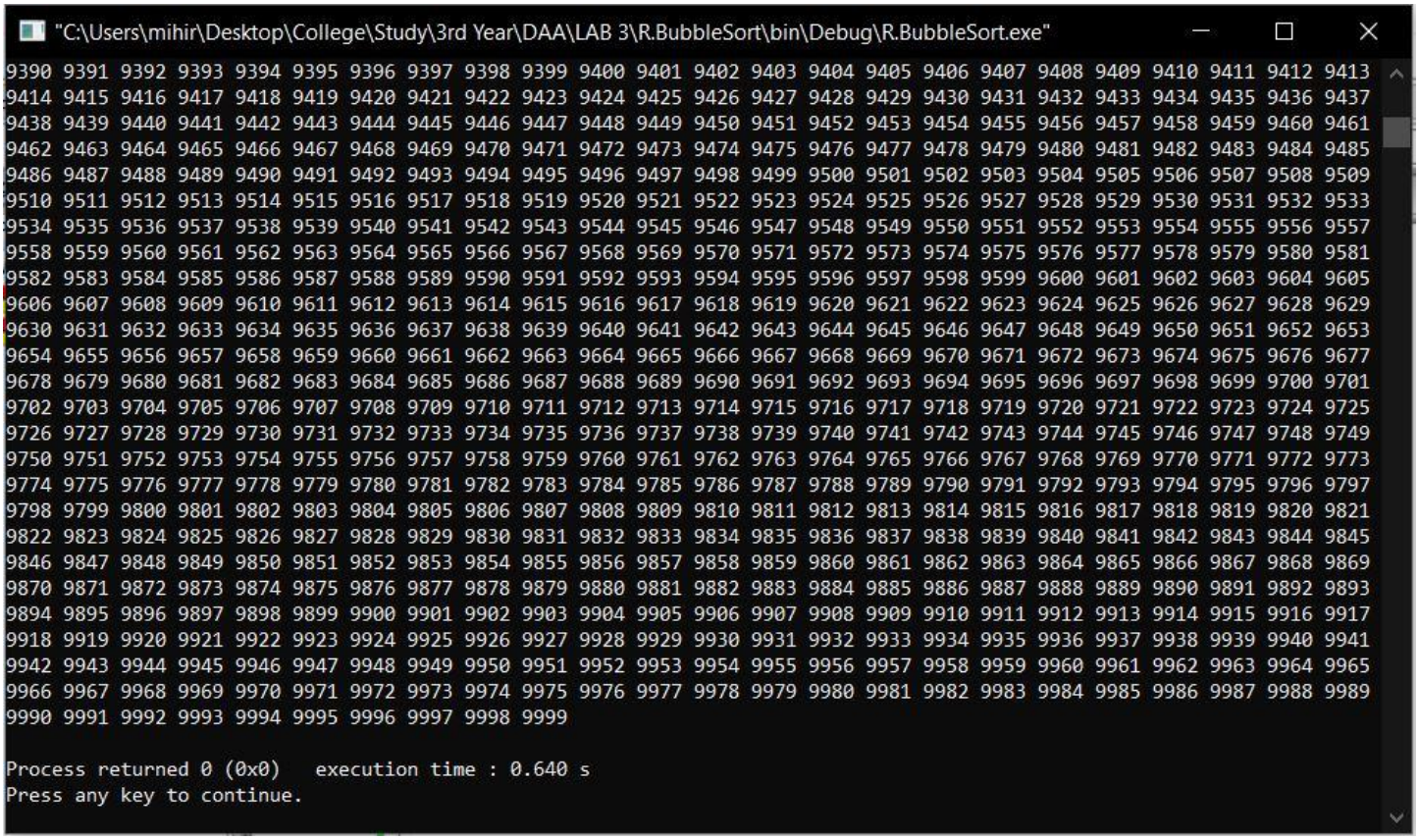
2. Recursive Bubble Sort

Code:

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

void BubbleSortRecursion(int a[],int num)
{
int i,j,temp;

i=num;
if(i>0)
{
for(j=0; j<num-1; j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
BubbleSortRecursion(a,num-1);
}
else
{
return;
```

```
}
}

void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", arr[i]);
printf("\n");

}

int main()
{
int no=10000;
int j = no;
int arr[no];
for(int i=0;i<no;i++)
{
arr[i]=no-j;
j--;
}
int n = sizeof(arr)/sizeof(arr[0]);
BubbleSortRecursion(arr, n);
printf("Sorted array: \n");
printArray(arr, n);
return 0;
```

```
}
```

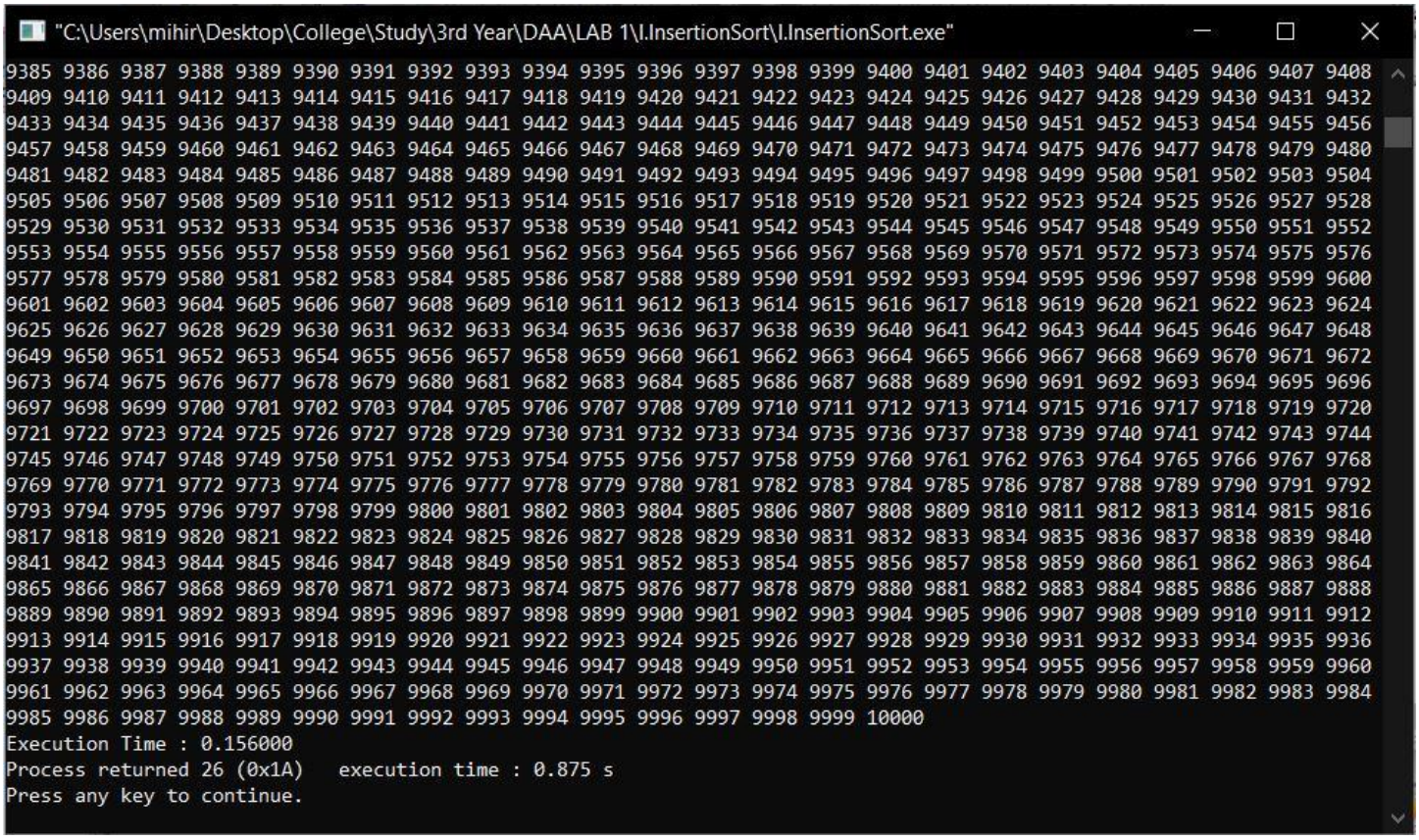

"C:\Users\mihir\Desktop\College\Study\3rd Year\DAA\LAB 3\R.BubbleSort\bin\Debug\R.BubbleSort.exe"
9390 9391 9392 9393 9394 9395 9396 9397 9398 9399 9400 9401 9402 9403 9404 9405 9406 9407 9408 9409 9410 9411 9412 9413
9414 9415 9416 9417 9418 9419 9420 9421 9422 9423 9424 9425 9426 9427 9428 9429 9430 9431 9432 9433 9434 9435 9436 9437
9438 9439 9440 9441 9442 9443 9444 9445 9446 9447 9448 9449 9450 9451 9452 9453 9454 9455 9456 9457 9458 9459 9460 9461
9462 9463 9464 9465 9466 9467 9468 9469 9470 9471 9472 9473 9474 9475 9476 9477 9478 9479 9480 9481 9482 9483 9484 9485
9486 9487 9488 9489 9490 9491 9492 9493 9494 9495 9496 9497 9498 9499 9500 9501 9502 9503 9504 9505 9506 9507 9508 9509
9510 9511 9512 9513 9514 9515 9516 9517 9518 9519 9520 9521 9522 9523 9524 9525 9526 9527 9528 9529 9530 9531 9532 9533
9534 9535 9536 9537 9538 9539 9540 9541 9542 9543 9544 9545 9546 9547 9548 9549 9550 9551 9552 9553 9554 9555 9556 9557
9558 9559 9560 9561 9562 9563 9564 9565 9566 9567 9568 9569 9570 9571 9572 9573 9574 9575 9576 9577 9578 9579 9580 9581
9582 9583 9584 9585 9586 9587 9588 9589 9590 9591 9592 9593 9594 9595 9596 9597 9598 9599 9600 9601 9602 9603 9604 9605
9606 9607 9608 9609 9610 9611 9612 9613 9614 9615 9616 9617 9618 9619 9620 9621 9622 9623 9624 9625 9626 9627 9628 9629
9630 9631 9632 9633 9634 9635 9636 9637 9638 9639 9640 9641 9642 9643 9644 9645 9646 9647 9648 9649 9650 9651 9652 9653
9654 9655 9656 9657 9658 9659 9660 9661 9662 9663 9664 9665 9666 9667 9668 9669 9670 9671 9672 9673 9674 9675 9676 9677
9678 9679 9680 9681 9682 9683 9684 9685 9686 9687 9688 9689 9690 9691 9692 9693 9694 9695 9696 9697 9698 9699 9700 9701
9702 9703 9704 9705 9706 9707 9708 9709 9710 9711 9712 9713 9714 9715 9716 9717 9718 9719 9720 9721 9722 9723 9724 9725
9726 9727 9728 9729 9730 9731 9732 9733 9734 9735 9736 9737 9738 9739 9740 9741 9742 9743 9744 9745 9746 9747 9748 9749
9750 9751 9752 9753 9754 9755 9756 9757 9758 9759 9760 9761 9762 9763 9764 9765 9766 9767 9768 9769 9770 9771 9772 9773
9774 9775 9776 9777 9778 9779 9780 9781 9782 9783 9784 9785 9786 9787 9788 9789 9790 9791 9792 9793 9794 9795 9796 9797
9798 9799 9800 9801 9802 9803 9804 9805 9806 9807 9808 9809 9810 9811 9812 9813 9814 9815 9816 9817 9818 9819 9820 9821
9822 9823 9824 9825 9826 9827 9828 9829 9830 9831 9832 9833 9834 9835 9836 9837 9838 9839 9840 9841 9842 9843 9844 9845
9846 9847 9848 9849 9850 9851 9852 9853 9854 9855 9856 9857 9858 9859 9860 9861 9862 9863 9864 9865 9866 9867 9868 9869
9870 9871 9872 9873 9874 9875 9876 9877 9878 9879 9880 9881 9882 9883 9884 9885 9886 9887 9888 9889 9890 9891 9892 9893
9894 9895 9896 9897 9898 9899 9900 9901 9902 9903 9904 9905 9906 9907 9908 9909 9910 9911 9912 9913 9914 9915 9916 9917
9918 9919 9920 9921 9922 9923 9924 9925 9926 9927 9928 9929 9930 9931 9932 9933 9934 9935 9936 9937 9938 9939 9940 9941
9942 9943 9944 9945 9946 9947 9948 9949 9950 9951 9952 9953 9954 9955 9956 9957 9958 9959 9960 9961 9962 9963 9964 9965
9966 9967 9968 9969 9970 9971 9972 9973 9974 9975 9976 9977 9978 9979 9980 9981 9982 9983 9984 9985 9986 9987 9988 9989
9990 9991 9992 9993 9994 9995 9996 9997 9998 9999
Process returned 0 (0x0)   execution time : 0.640 s
Press any key to continue.

3. Iterative Insertion Sort

Code:

```
#include<stdio.h>
#include<time.h>
void main()
{
   int n=10000;
   clock_t t;
   printf("Itterative Insertion Sort\n");
   printf("\nEnter number of elements : ");
   //scanf("%d",&n);
   //printf("\nEnter array : ");
   int arr[n];
   for(int i=0; i<n; i++)
   {
      arr[i] = 10000-i;
   }
   t = clock();
   for(int i=0; i<n-1; i++)
   {
      for(int j=i; j>=0; j--)
      {
         if(arr[j+1]<arr[j])
         {
            int temp;
            temp = arr[j+1];
```

```
            arr[j+1]=arr[j];
            arr[j] = temp;
        }
    }
  }
  t=clock()-t;
  for(int i=0; i<n; i++)
  {
    printf("%d ",arr[i]);
  }
  double time_taken = ((double)t)/CLOCKS_PER_SEC;
  printf("\nExecution Time : %f", time_taken);
}
```



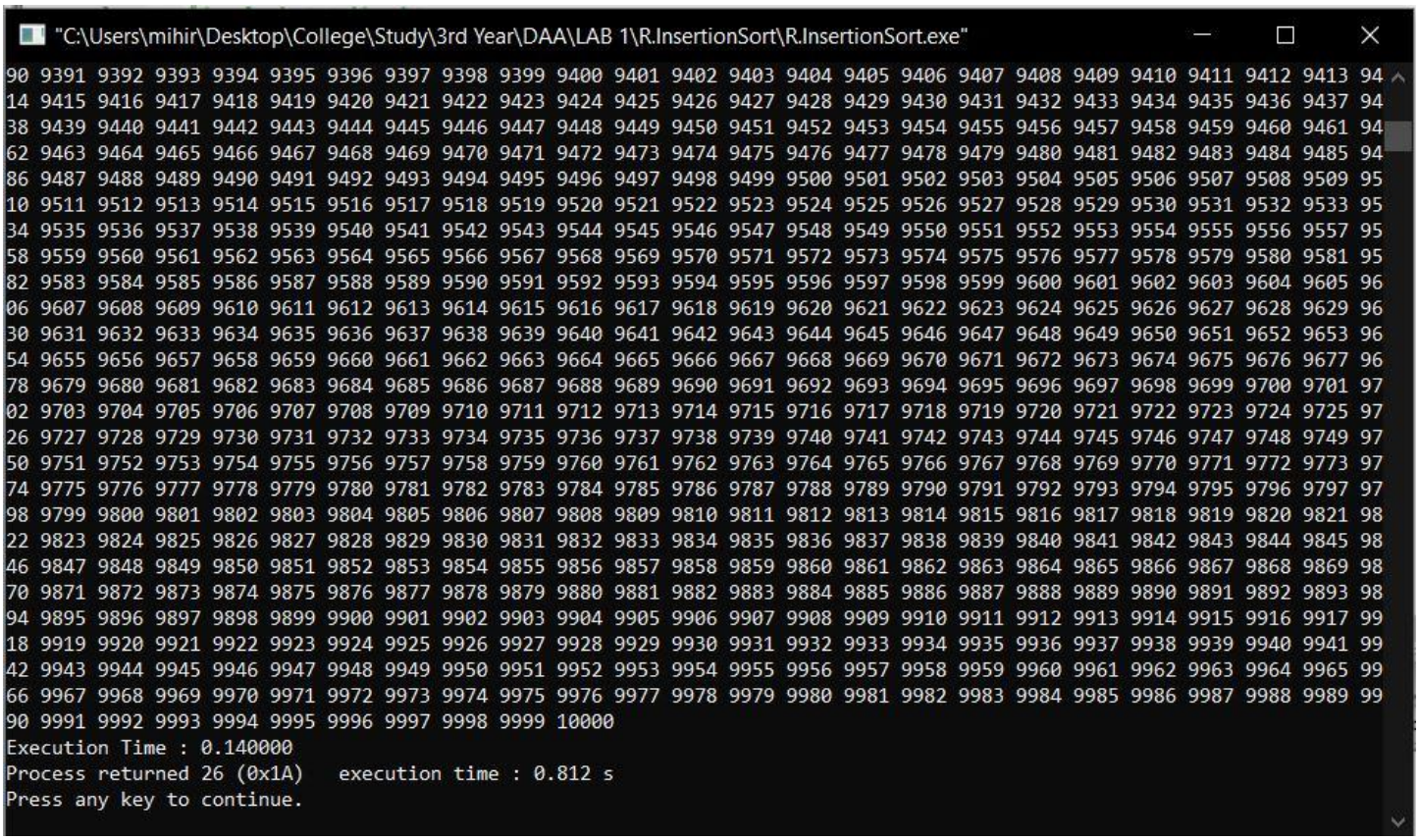
4. Recursive Insertion Sort

Code:

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

void RecSelectionSort(int arr[], int num, int n)
{
   int i = num;
   if(n<0)
      return;
   for(int j=i; j>=0; j--)
   {
      if(arr[j+1]<arr[j])
      {
         int temp;
         temp = arr[j+1];
         arr[j+1]=arr[j];
         arr[j] = temp;
      }
   }
   RecSelectionSort(arr,num+1,n-1);
```

```
}

void main()
{
   int n=10000;
   printf("Recursive Insertion Sort\n");
   //printf("\nEnter number of elements : ");
   //scanf("%d",&n);
   //printf("\nEnter array : ");
   int arr[n];
   for(int i=0; i<n; i++)
   {
      arr[i]=10000-i;
   }

   int num=0;
   clock_t t;
   t = clock();
   RecSelectionSort(arr, num, n);
   t = clock() - t;
   double time_taken = ((double)t)/CLOCKS_PER_SEC; // in seconds
```

```
    for(int i=0; i<n; i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\nExecution Time : %f", time_taken);
}
```

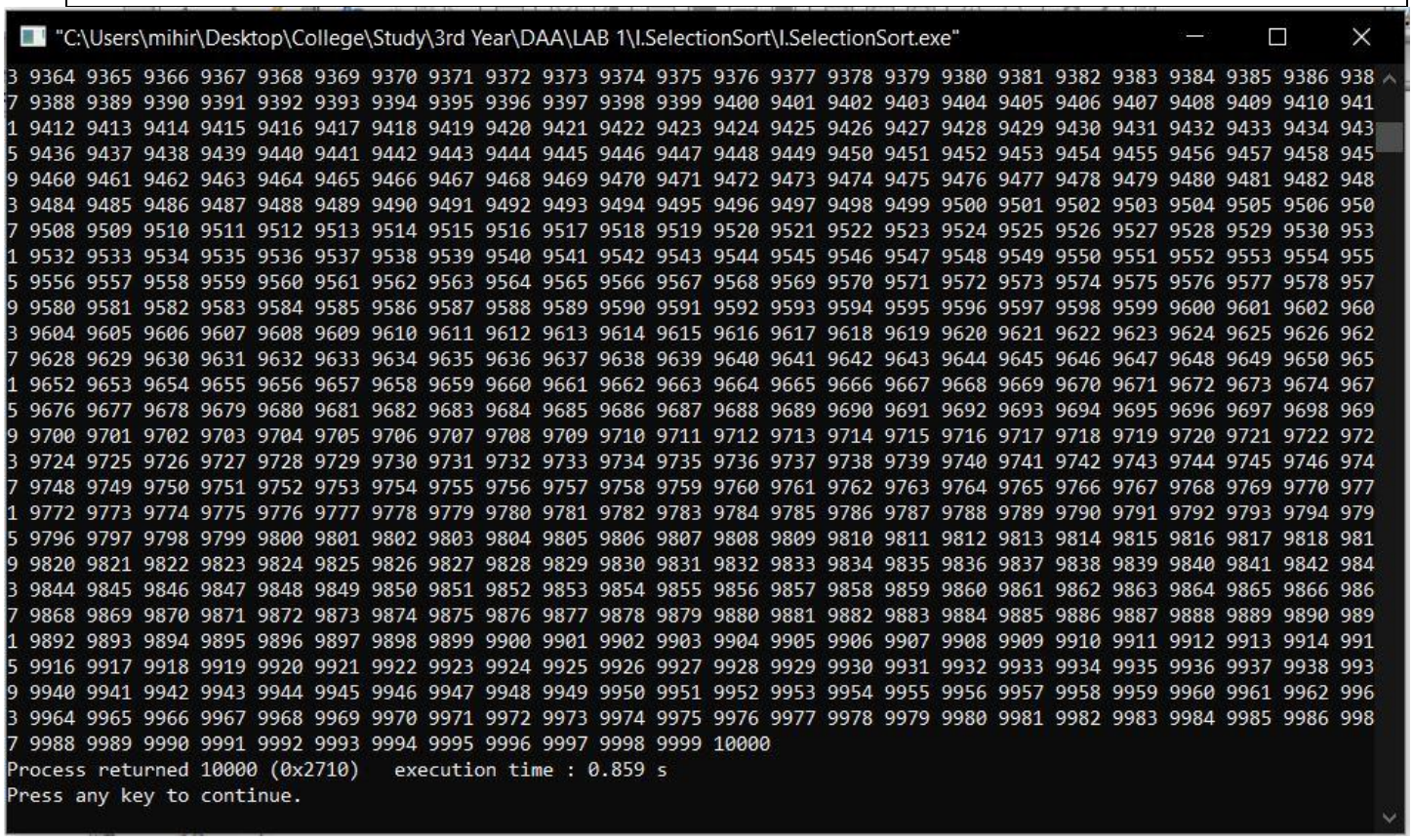


5. Iterative Selection Sort

Code:

```
#include<stdio.h>

void main()
{
    int n=10000;
    printf("Itterative Selection Sort\n");
    printf("\nEnter number of elements : ");
    //scanf("%d",&n);
    printf("\nEnter array : ");
    int arr[n];
    for(int i=0; i<n; i++)
    {
        arr[i] = 10000-i;
    }

    int index;
    for(int i=0; i<n-1; i++)
    {
        index = i;
        for(int j=i+1; j<n; j++)
        {
            if(arr[index]>arr[j])
            {
                index = j;
```

```
        }
    }

    int temp;
    temp = arr[i];
    arr[i] = arr[index];
    arr[index] = temp;
  }

  for(int i=0; i<n; i++)
  {
    printf("%d ",arr[i]);
  }
}
```

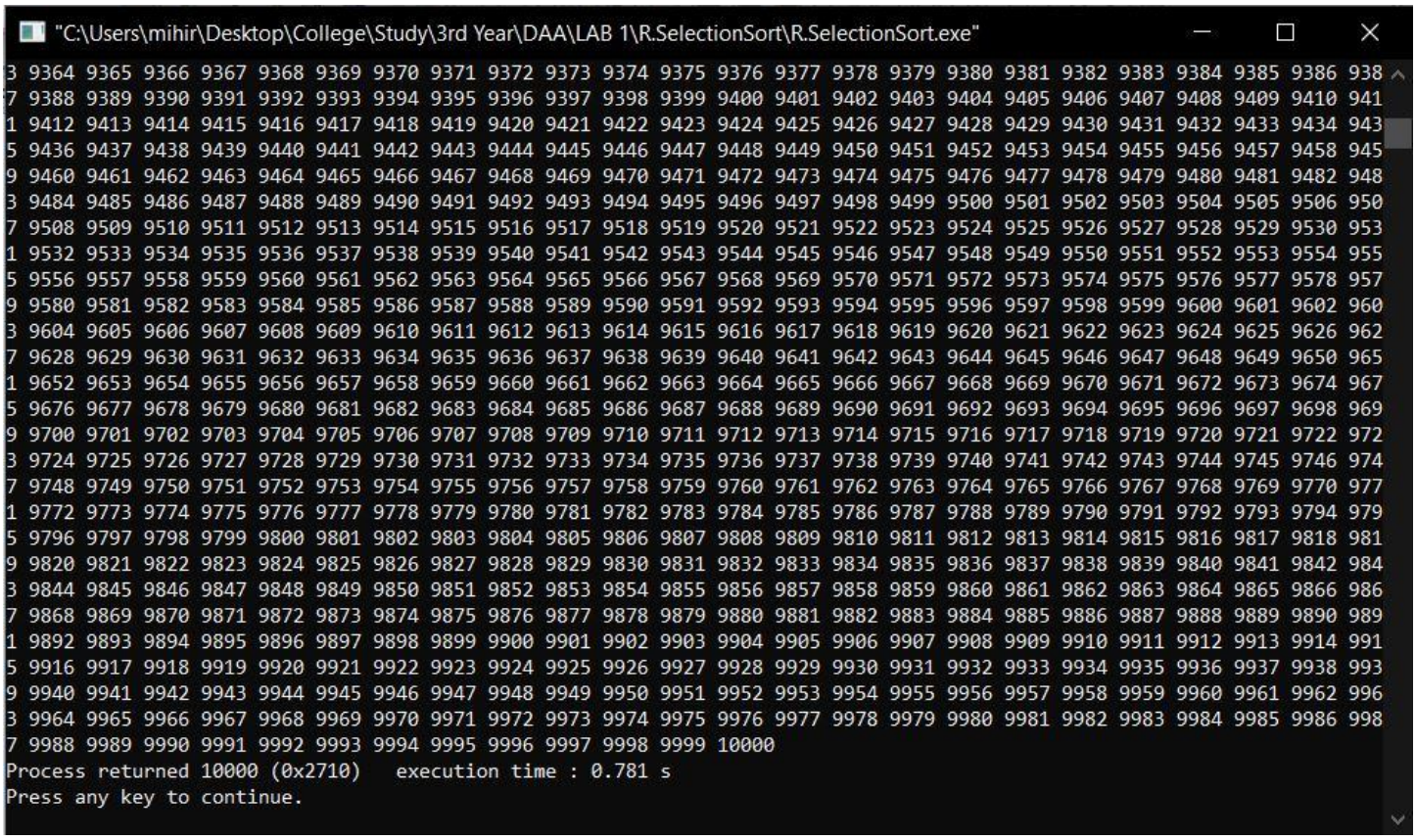


6. Recursive Selection Sort

Code:

```
#include<stdio.h>

void RecSelectionSort(int arr[], int num, int n)
{
    int index,i;
    i=num;
    if(n<0)
        return;
    index = i;
    for(int j=i+1; j<n; j++)
    {
        if(arr[index]>arr[j])
        {
            index = j;
        }
    }

    int temp;
    temp = arr[i];
    arr[i] = arr[index];
    arr[index] = temp;

    RecSelectionSort(arr,num+1,n-1);
```

```
}

void main()
{
    int n=10000;
    printf("Recursive Selection Sort\n");
    printf("\nEnter number of elements : ");
    //scanf("%d",&n);
    printf("\nEnter array : ");
    int arr[n];
    for(int i=0; i<n; i++)
    {
        arr[i] = 10000-i;
    }
    int num=0;
    RecSelectionSort(arr, num, n);
    for(int i=0; i<n; i++)
    {
        printf("%d ",arr[i]);
    }
}
```



Observation Table:

FOR N=10,000

	Iterative	Recursive
Bubble Sort	0.689 s	0.640 s
Insertion Sort	0.875 s	0.812 s
Selection Sort	0.859 s	0.781 s