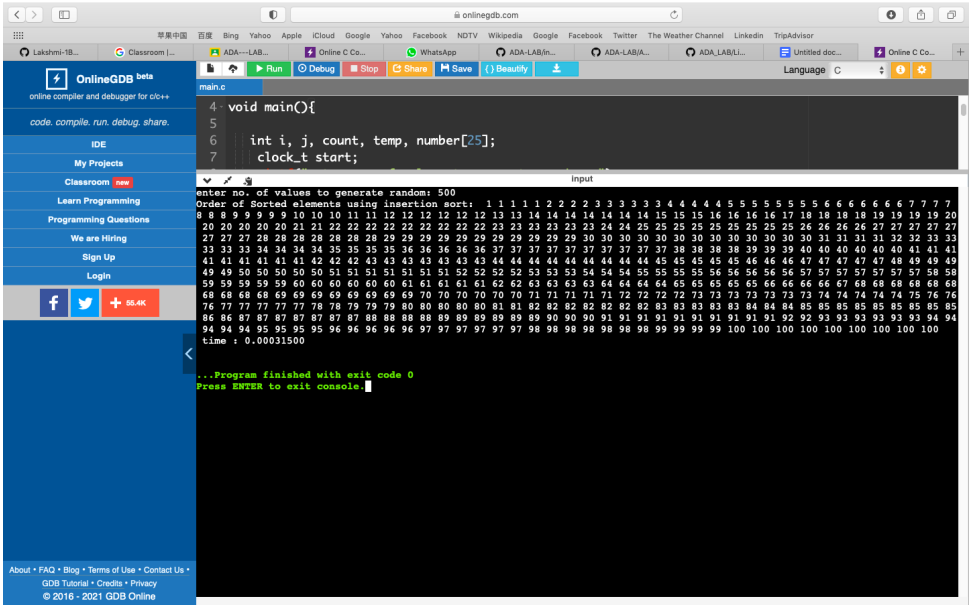
LAKSHMI S KUMAR
1BM19CS078

```
#include
<stdio.h>


int right = 1;

int left = 0;


int search(int a[], int n, int mobile)

{

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

       if (a[i] == mobile)

          return i + 1; // returning the position of the highest
mobile element
}




int getMobileElement(int a[], int dir[], int n) // to find the
largest mobile integer
{

   int mobile_prev = 0, mobile = 0;

   for (int i = 0; i < n; i++)
```

```
    {
        // direction 0 represents left
        if (dir[a[i]-1] == left && i!=0)
        {
            if (a[i] > a[i-1] && a[i] > mobile_prev)
            {
                mobile = a[i];
                mobile_prev = mobile;
            }
        }

        // direction 1 represents right
        if (dir[a[i]-1] == right && i!=n-1)
        {
            if (a[i] > a[i+1] && a[i] > mobile_prev)
            {
                mobile = a[i];
                mobile_prev = mobile;
            }
        }
    }

    if (mobile == 0 && mobile_prev == 0)
        return 0;
    else
        return mobile;
}

void swap(int* a, int* b) {
    int t = *a;
    *a=*b;
    *b=t;
```

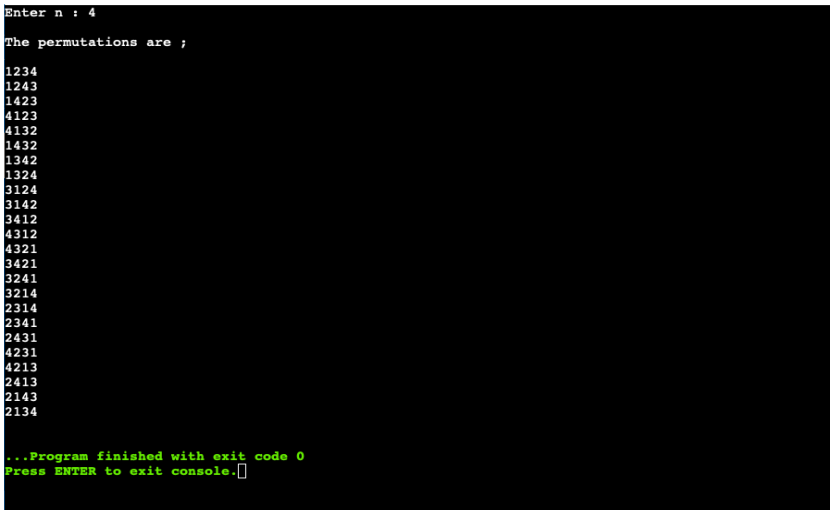
```
}
int printOnePermutation(int a[], int dir[], int n)
{
   int mobile = getMobileElement(a, dir, n);
   int pos = search(a, n, mobile);



   if (dir[a[pos - 1] - 1] ==  left)
      swap(&a[pos-1], &a[pos-2]);

   else if (dir[a[pos - 1] - 1] == right)
      swap(&a[pos], &a[pos-1]);

   // changing the directions for elements greater than largest
mobile integer
   for (int i = 0; i < n; i++)
   {
       if (a[i] > mobile)
       {
           if (dir[a[i] - 1] == right)
               dir[a[i] - 1] = left;
           else if (dir[a[i] - 1] == left)
               dir[a[i] - 1] = right;
       }
   }

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

```
int fact(int n)
{
   int res = 1;
   for (int i = 1; i <= n; i++)
       res = res * i;
   return res;
}



// one by one prints all permutations
void onebyonePermutation(int n)
{
```



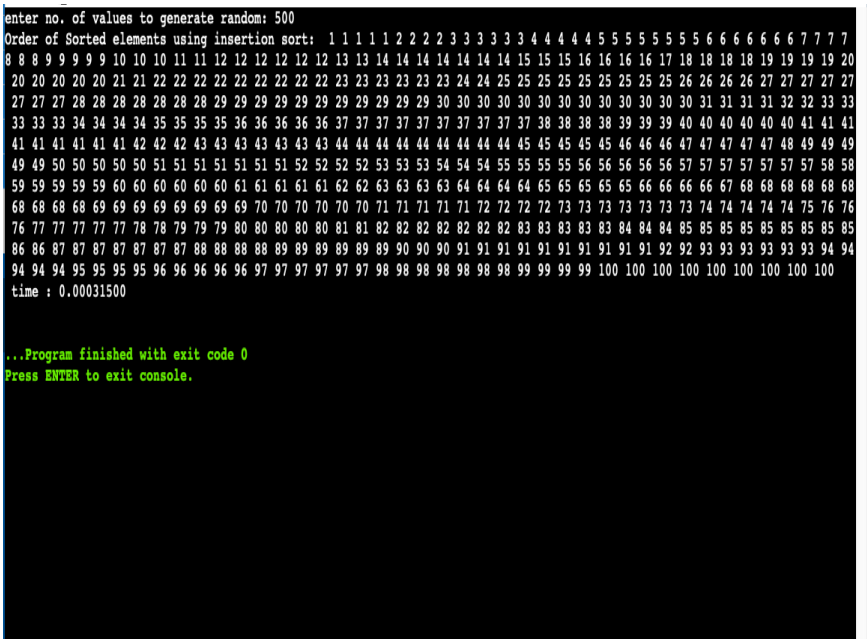

```
   int a[n];//to store current permutation

   int dir[n];//to store current directions

   // storing the elements from 1 to n and printing first
permutation.
   for (int i = 0; i < n; i++)
```

```
    {
        a[i] = i + 1; //ith element will be i+1, a[0] will be 1
        printf("%d", a[i]);
    }
    printf("\n");

    //direction is initialised to left
    for (int i = 0; i < n; i++)
        dir[i] =  left;

    // for generating permutations in the order, (n)! -1 number of
times.
    for (int i = 1; i < fact(n); i++)
        printOnePermutation(a, dir, n);
}

int main()
{
    int n;
    printf("Enter n : ");
    scanf("%d", &n);
    printf("\nThe permutations are ;\n\n");
    onebyonePermutation(n);
    return 0;
}
```




```
#include
<stdio.h>
                    #include <time.h>
                    #include<stdlib.h>
                    void main(){


                      int i, j, count, temp, number[25];
                       clock_t start;
                      printf("enter no. of values to generate random: ");
                      scanf("%d",&count);



                      for(i=0;i<count;i++)
                         number[i]=rand()%100 +1;


                       start=clock();
                      for(i=1;i<count;i++){
                         temp=number[i];
                         j=i-1;
                         while((temp<number[j])&&(j>=0)){
```

```
        number[j+1]=number[j];
        j=j-1;
      }
     number[j+1]=temp;
  }
   start=clock()-start;
   double t=((double)start)/CLOCKS_PER_SEC;
  printf("Order of Sorted elements using insertion sort: ");
  for(i=0;i<count;i++)
     printf(" %d",number[i]);
         printf("\n time : %.8f\n",t);


}
```



```
  }
   start=clock()-start;
   double t=((double)start)/CLOCKS_PER_SEC;
  printf("Order of Sorted elements using insertion sort: ");
  for(i=0;i<count;i++)
     printf(" %d",number[i]);
         printf("\n time : %.8f\n",t);


}
```