

# 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]);

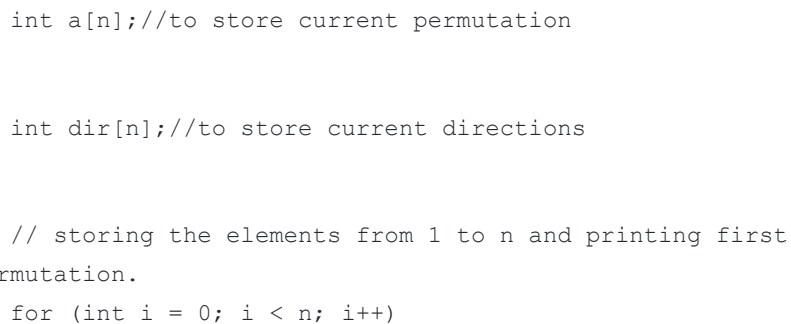
        // changing the directions for elements greater than largest
        mobile integer
        for (int i = 0; i < n; i++)
        {
            if (a[i] > mobile)
            {
                if (dir[a[i] - 1] == right)
                    dir[a[i] - 1] = left;
                else if (dir[a[i] - 1] ==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

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

        printf("\n");
    }

```

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



```

    {
        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;
}

```

```
Enter n : 4
The permutations are ;
1234
1243
1423
4123
4132
1432
1342
1324
3124
3142
3412
4312
4321
3421
3241
3214
2314
2341
2431
4231
4213
2413
2143
2134
...Program finished with exit code 0
Press ENTER to exit console.
```

```
#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);

}

```

```

enter no. of values to generate random: 500
Order of Sorted elements using insertion sort: 1 1 1 1 1 2 2 2 2 3 3 3 3 3 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7
8 8 8 9 9 9 9 10 10 10 11 11 12 12 12 12 12 13 13 14 14 14 14 14 15 15 15 16 16 16 17 18 18 18 19 19 19 20
20 20 20 20 20 21 21 22 22 22 22 22 22 22 22 23 23 23 23 23 24 24 25 25 25 25 25 25 25 26 26 26 27 27 27 27
27 27 27 28 28 28 28 28 28 29 29 29 29 29 29 29 29 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 32 32 33 33
33 33 33 34 34 34 35 35 35 36 36 36 36 37 37 37 37 37 37 37 37 38 38 38 39 39 39 40 40 40 40 40 41 41 41
41 41 41 41 41 42 42 42 43 43 43 43 43 44 44 44 44 44 44 44 45 45 45 45 46 46 46 47 47 47 47 47 48 49 49 49
49 49 50 50 50 50 51 51 51 51 51 51 52 52 52 53 53 53 54 54 54 55 55 55 56 56 56 56 57 57 57 57 57 58 58
59 59 59 59 60 60 60 60 61 61 61 61 62 62 63 63 63 64 64 64 65 65 65 65 66 66 66 66 67 68 68 68 68 68
68 68 68 69 69 69 69 69 70 70 70 70 70 71 71 71 71 72 72 72 73 73 73 73 73 74 74 74 74 75 76 76
76 77 77 77 77 77 78 78 79 79 80 80 80 81 81 82 82 82 82 82 83 83 83 83 84 84 84 85 85 85 85 85 85
86 86 87 87 87 87 87 88 88 88 89 89 89 89 90 90 90 91 91 91 91 91 91 91 92 92 93 93 93 93 93 94 94
94 94 95 95 95 96 96 96 97 97 97 97 97 98 98 98 98 98 99 99 99 100 100 100 100 100 100 100 100
time : 0.00031500

...Program finished with exit code 0
Press ENTER to exit console.

```

```
}  
  
    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);  
  
}
```