S.No: 1	Exp. Name: ***Maximum Element Of Array***	Date:2024-08-12

ID: 0231CYS014

Aim:

Implement a program to find the maximum element in an array of size n.

Input Format:

- The first line of the input represents an integer n
- The second line of the input represents n space separated elements of the array (Integers)

Output Format:

- The first line of the output represents the input array.
- The second line contains an integer representing the maximum element of the array.

Source Code:

CTC37586.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i =0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   int max = arr[0];
   printf("[");
   for(int i=0;i<n;i++){
      if(i<n-1){
         printf("%d, ",arr[i]);
      }
      else
         printf("%d",arr[i]);
   }
   printf("]");
   for(int i = 1;i<n;i++){
      if(arr[i]>max){
         max = arr[i];}
      }
   printf("\n");
   printf("%d",max);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
4
7 2 9 5
[7, 2, 9, 5]
9

Test Case - 2
User Output
3
-1 -4 0
[−1, −4, 0]
0

S.No: 2	Exp. Name: ***Sum of All Elements in an Array***	Date:2024-08-12

ID: 0231CYS014

Aim:

Implement a program to find the sum of all elements in an array of size n.

Input Format:

- The first line of the input represents an integer n
- The second line of the input represents n space separated elements of the array (Integers)

Output Format:

- The first line of the output represents the input array.
- The second line of the output contains an integer representing the sum of all element of the array.

Source Code:

CTC37596.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i =0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   printf("[");
   for(int i=0;i<n;i++){
      if(i<n-1){
         printf("%d, ",arr[i]);
      }
      else
         printf("%d]\n",arr[i]);
   }
   int sum = 0;
   for(int i=0;i<n;i++){
      sum += arr[i];
   }
   printf("%d",sum);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
3
1 2 3
[1, 2, 3]
6

Test Case - 2
User Output
5

4 -2 7 1 0
[4, −2, 7, 1, 0]
10

S.No: 3	Exp. Name: ***Reverse the Elements of an Array***	Date:2024-08-14

ID: 0231CYS014

Aim:

Implement a program to reverse the elements of an array of size n.

Input Format:

- The first line of the input contains an integer n representing the number of elements.
- The second line of the input represents n space separated elements of the array (Integers).

Output Format:

- The output represents the elements of the input array in reversed order.

Source Code:

CTC37597.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i = 0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   printf("[");
   for(int i =n-1;i>=0;i--){
      if(i > 0){
         printf("%d, ",arr[i]);
      }
      else
         printf("%d]",arr[i]);
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
3
1 2 3
[3, 2, 1]

Test Case - 2
User Output
6
0 -1 2 -3 4 -5
[-5, 4, -3, 2, -1, 0]

S.No: 4	Exp. Name: ***Check If Array Is Sorted in Ascending Order***	Date:2024-08-21

Aim:

Implement a program to check if an array of size n is sorted in ascending order or not.

Input Format:

- The first line of the input represents an integer n
- The second line of the input represents n space separated elements of the array (Integers)

Output Format:

- The output represents **True** if the array is sorted in ascending order, otherwise, it is **False**

Source Code:

CTC37601.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i = 0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   int count = 0;
   for(int i = 0;i<n-1;i++){
      for(int j=0;j<n-1-i;j++){
         if(arr[j]>arr[j+1]){
            int temp = arr[j];
            arr[j]= arr[j+1];
            arr[j+1]=temp;
            count = 1;
         }
      }
   }
   if(count == 0 ){
      printf("True\n");
   }
   else{
      printf("False\n");
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
1 2 3 4 5
True

Test Case - 2

User Output
4
10 5 15 20
False

S.No: 5	Exp. Name: ***Count Occurrence of an Element in an Array***	Date:2024-08-21

ID: 0231CYS014

Aim:

Implement a program to count the occurrence of a specific element k in an array of size n

Input Format:

- The first line of the input represents an integer n
- The second line of the input represents n space separated elements of the array (Integers)
- The third line of the input represents an integer k to search the occurrence in the array.

Output Format:

- The output contains an integer representing the count of the occurrence of integer k in the array.

Source Code:

CTC37615.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i = 0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   int k;
   int count=0;
   scanf("%d",&k);
   for(int i = 0;i<n;i++){
      if(arr[i]==k){
         count++;
      }
   }
   printf("%d",count);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
1 2 3 2 4
2
2

Test Case - 2
User Output
4
7 8 9 10
6
0

S.No: 6	Exp. Name: ***Creation And Traversal of 2D Array***	Date:2024-08-23

ID: 0231CYS014

Aim:

Implement a program that takes the number of rows r and columns c from the user to create a 2D array, and then traverses and prints the array in both row-major and column-major order.

Input Format:

- The first line of input contains two integers r and c separated by a space, representing the number of rows and columns, respectively.
- The second line of input contains $r \times c$ space-separated elements to populate the 2D array.

Output Format:

- The first r lines of the output should each contain c space-separated elements, representing the row-major order traversal of the array.
- The next c lines of the output should each contain r space-separated elements, representing the column-major order traversal of the array.

Source Code:

CTC37640.c

```
#include <stdio.h>
int main(){
   int r,c;
   scanf("%d%d",&r,&c);
   int arr[r][c];
   for(int i = 0;i<r;i++){
      for(int j = 0;j<c;j++){
         scanf("%d",&arr[i][j]);
      }
   }
   for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }
   for(int j = 0;j<c;j++){
      for(int i = 0;i<r;i++){
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

2 3
1 2 3 4 5 6
1 2 3
4 5 6
1 4
2 5
3 6

Test Case - 2
User Output
2 2
1 3 2 4
1 3
2 4
1 2
3 4

S.No: 7	Exp. Name: ***Transpose of a Given Matrix***	Date:2024-08-23

Aim:

Implement a program that takes the number of rows r and columns c from the user to create a matrix and prints both the original matrix and its transpose.

Input Format:

- The first line of input contains two integers r and c separated by a space, representing the number of rows and columns, respectively.
- The next r lines each contain c space-separated elements to populate the matrix.

Output Format:

- The first r lines of the output should each contain c space-separated elements, representing the original matrix.
- The next c lines of the output should each contain r space-separated elements, representing the transpose of the matrix.

Source Code:

CTC37642.c

```
#include <stdio.h>
int main(){
   int r,c;
   scanf("%d%d",&r,&c);
   int arr[r][c];
   for(int i = 0;i<r;i++){
      for(int j = 0;j<c;j++){
         scanf("%d",&arr[i][j]);
      }
   }
   for(int i = 0;i<r;i++){
      for(int j=0;j<c;j++){
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }
   for(int j=0;j<c;j++){
      for(int i=0;i<r;i++){
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
2 3
1 2 3
4 5 6

1 2 3
4 5 6
1 4
2 5
3 6

Test Case - 2
User Output
3 2
7 8
9 10
11 12
7 8
9 10
11 12
7 9 11
8 10 12

ID: 0231CYS014

S.No: 8	Exp. Name: ***Check If Matrix Is Sparse and Print Sparse Representation***	Date:2024-08-23

Aim:

Implement a program that determines whether a given matrix is sparse and, if so, prints its non-zero elements. A matrix is considered sparse if the number of zero elements is greater than the number of non-zero elements.

Input Format:

- The first line of input contains two integers r and c separated by a space, representing the number of rows and columns, respectively.
- The next r lines each contain c space-separated integers, representing the elements of the matrix.

Output Format:

- The first line of the output should be a boolean value (True or False), indicating whether the matrix is sparse.
- If the matrix is sparse, print the non-zero elements in the format row column value, each on a new line. If the matrix is not sparse, print nothing.

Source Code:

CTC37649.c

```
#include <stdio.h>
int main(){
   int r,c,i,j;
   scanf("%d%d",&r,&c);
   int arr[r][c];
   for(i=0;i<r;i++){
      for(j=0;j<c;j++){
         scanf("%d",&arr[i][j]);
      }
   }
   int z=0,nz=0;
   for(i=0;i<r;i++){
      for(j=0;j<c;j++){
         if(arr[i][j]!=0){
            nz++;
         }
         else{
            z++;
         }
      }
   }
   if(nz>z){
      printf("False\n");
   }
   else{
      printf("True\n");
      int spars[nz][3],k=0;
      for(i=0;i<r;i++){
         for(j=0;j<c;j++){
            if(arr[i][j]!=0){
               spars[k][0]=i;
               spars[k][1]=j;
```

```
                spars[k][2]=arr[i][j];
                k++;
             }
          }
       }
       for(i=0;i<nz;i++){
          for(j=0;j<3;j++){
             if(j<2){
                printf("%d ",spars[i][j]);
             }
             else{
                printf("%d\n",spars[i][j]);
             }
          }
       }
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
3 3
0 0 3
0 0 6
0 0 0
True
0 2 3
1 2 6

Test Case - 2
User Output
3 3
1 2 3
4 0 6
7 8 9
False

ID: 0231CYS014

S.No: 9	Exp. Name: ***Triplex Representation***	Date:2024-08-25

ID: 0231CYS014

Aim:

Imagine you are working on a project for a data analytics company. They have a dataset that consists of a large matrix where each cell represents a measurement taken at different times and locations. This matrix is quite large, and most of its elements are zeros, which means it is a sparse matrix

Your task is to develop a tool that will help the company analyze this matrix efficiently.

Matrix Input:

- Create a tool that allows the user to input the dimensions of the matrix (number of rows and columns) and then enter the elements of the matrix. Each element is an integer.

Matrix Analysis:

- Implement functionality to determine if the matrix is sparse. A matrix is considered sparse if more than half of its elements are zeros.

Sparse Matrix Representation:

- If the matrix is identified as sparse, convert it to its triplet form representation. The triplet form should consist of a list of tuples where each tuple represents a non-zero element in the format (row_index, column_index, value).

Output Requirements:

- If the matrix is sparse, print its triplet representation.

Input Format

- First Line: Two integers r and c representing the number of rows and columns of the matrix separated by a space
- Following Lines: r rows represents c elements with a new line for every **i^{th}** row where the elements are separated by a space

Output Format

- Triplet Representation (if sparse): Print the triplet representation of the matrix. Each triplet should be printed on a new line in the format (row_index, column_index, value).
- if not sparse : **Return -1**

Constraints

- 1 <= rows, cols <= 1000
- -10^6 <= matrix[i][j] <= 10^6 (matrix elements range between -1,000,000 and 1,000,000)
- The input matrix will be a rectangular grid (i.e., all rows have the same number of columns).

Source Code:

CTC38898.c

```
#include <stdio.h>
int main(){
   int r,c,i,j;
   scanf("%d%d",&r,&c);
   int arr[r][c];
   for(i=0;i<r;i++){
```

```
      for(j=0;j<c;j++){
         scanf("%d",&arr[i][j]);
      }
   }
   int nm =r*c;
   int nz=0,z=0;
   for(i=0;i<r;i++){
      for(j=0;j<c;j++){
         if(arr[i][j]==0){
            z++;
         }
         else{
            nz++;
         }
      }
   }
   int spp[nz][3];
   int k=0;
      for(i=0;i<r;i++){
         for(j=0;j<c;j++){
            if(arr[i][j]!=0){
               spp[k][0]=i;
               spp[k][1]=j;
               spp[k][2]=arr[i][j];
               k++;
            }
         }
      }
   if(nz==0){
      printf("-1");
   }
   else if(z>nz){
      for(int i=0;i<nz;i++){
         printf("%d %d %d\n",spp[i][0],spp[i][1],spp[i][2]);
      }
   }
   else{
      printf("-1");
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
2 2
0 0
0 0
-1

Test Case - 2
User Output
3 3

0 1 0
2 0 3
0 4 0
0 1 1
1 0 2
1 2 3
2 1 4

S.No: 10	Exp. Name: ***Linear search Technique***	Date:2024-08-24

ID: 0231CYS014

Aim:

Write a program to check whether the given element is present or not in the array of elements using linear search.

Input format:

- The first line of input contains an integer representing the no. of elements of the array.
- The second line of input contains the array of integers separated by space.
- The last line of input contains the key element to be searched.

Output format:

- If the element is found, print the index.
- If the element is not found, print **Not found**.

Sample Test Case:

Input:

7
1 2 3 4 3 5 6
3

Output:

2

Source Code:

CTC17129.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n],i,k,l,count=0;
   for(i=0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   scanf("%d",&k);
   for(i=0;i<n;i++){
      if(arr[i]==k){
         printf("%d",i);
         break;
      }
      else if(i==n-1){
         printf("Not found");
      }
   }
   // if(count==1){
   //    printf("%d",l);
   // }
   // else{
   //    printf("Not found");
   // }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
7
1 2 3 4 3 5 6
3
2

Test Case - 2
User Output
10
1 2 3 4 5 6 7 8 9 19
20
Not found

ID: 0231CYS014

Noida Institute of Engineering and Technology

2023-2027-CYS

S.No: 11	Exp. Name: ***Binary Search***	Date:2024-08-25

Aim:

Given a list of N numbers a1, a2, a3.....an. Find The Position of number **x** in the given list using binary search.

Note:
The default position of first element is 0, and find the element's position in the sorted order of occurrence. Print -1 if not found.

For repeated elements find the position of the last occurrence.

Constraints:
$0 < N < 1000$
$0 <$ element of array < 100
$0 < x < 100$

Input Format:
The first line takes the input value of **N.**
The second line takes input N space-separated integer values.
The third line takes the input value of **x** which needs to be searched.

Output Format:
Output is an integer that represents the position of **x** in the given list. Otherwise print -1

Source Code:

CTC14028.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n],i,j,x,index,pos;
   for(i=0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   scanf("%d",&x);
   i=0,j=n-1;
   int flag=0;
   while(i<=j){
      int mid =(i+j)/2;
      if(arr[mid]==x){
         index=mid;
         flag=1;
         // pos = mid+1;
         break;
      }
      else if(arr[mid]>x){
         j = mid-1;
      }
      else if(arr[mid]<x){
         i=mid+1;
      }
```

```
    }
    if(flag ==1){
       printf("%d",index);
    }
    else{
       printf("-1");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
1 5 8 7 4
8
2

Test Case - 2
User Output
5
12 85 97 45 2
145
-1

ID: 0231CYS014

S.No: 12	Exp. Name: ***Selection Sort Algorithm***	Date:2024-08-25

ID: 0231CYS014

Aim:

You are given a task to sort an array of integers using the Selection Sort algorithm. Your program should follow these specifications:

Input Format:

- The first line contains a single integer n, which represents the number of elements in the array.
- The second line contains n space-separated integers, which are the elements of the array.

Output Format:

- Print the sorted array on a single line, with elements separated by spaces.
- If the number of elements does not match n, print -1 and do not perform any sorting.

Constraints:

- The number of elements n is between 1 and 10^3.
- Each integer in the array is between -10^6 and 10^6.

Source Code:

CTC38968.c

```
#include <stdio.h>
int main(){
   int n,i,j,elements =0;
   scanf("%d",&n);
   int arr[n];
   for(i=0;;i++){
      scanf("%d",&arr[i]);
      elements++;
      char c;
      scanf("%c",&c);
      if(c =='\n'){
         break;
      }
   }
   for(i=0;i<n-1;i++){
      for(j=i+1;j<n;j++){
         if(arr[i]>arr[j]){
            int temp;
            temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
         }
         else{
            continue;
         }
      }
   }
   if(n==elements){
      for(i=0;i<n;i++){
         if(i<n-1){
            printf("%d ",arr[i]);
```

```
            }
            else{
               printf("%d\n",arr[i]);
            }
         }
      }
      else{
         printf("-1");
      }
   }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
6
10 5 3 8 6 7
3 5 6 7 8 10

Test Case - 2
User Output
3
2 4 6 8
-1

ID: 0231CYS014

S.No: 13	Exp. Name: ***Bubble Sort Algorithm***	Date:2024-08-25

ID: 0231CYS014

Aim:

You are given a task to sort an array of integers using the Bubble Sort algorithm. Your program should follow these specifications:

Input Format:

- The first line contains a single integer **n**, which represents the number of elements in the array.
- The second line contains **n** space-separated integers, which are the elements of the array.

Output Format:

- Print the sorted array on a single line, with elements separated by spaces.
- If the number of elements does not match **n**, print **-1** and do not perform any sorting.

Constraints:

- The number of elements n is between 1 and 1000.
- Each integer in the array is between -1,000,000 and 1,000,000.

Source Code:

CTC38978.c

```
#include <stdio.h>
int main(){
   int elements=0;
   int n;
   scanf("%d",&n);
   int arr[n];
   for(int i =0;;i++){
      scanf("%d",&arr[i]);
      elements++;
      char c;
      scanf("%c",&c);
      if(c=='\n'){
         break;
      }
   }
   for(int i =0;i<n-1;i++){
      for(int j = 0;j<n-1-i;j++){
         if(arr[j]>arr[j+1]){
            int temp = arr[j];
            arr[j]= arr[j+1];
            arr[j+1]= temp;
         }
      }
   }
   if(n==elements){
      for(int i =0;i<n;i++){
         if(i<n-1){
            printf("%d ",arr[i]);
         }
         else{
            printf("%d",arr[i]);
```

```
        }
      }
   }
   else{
      printf("-1");
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
6
10 5 3 8 6 7
3 5 6 7 8 10

Test Case - 2
User Output
3
2 4 6 8
-1

ID: 0231CYS014

S.No: 14	Exp. Name: ***Insertion Sort Algorithm***	Date:2024-08-26

ID: 0231CYS014

Aim:

You are given a task to sort an array of integers using the Insertion Sort algorithm. Your program should follow these specifications:

Input Format:

- The first line contains a single integer **n**, which represents the number of elements in the array.
- The second line contains **n** space-separated integers, which are the elements of the array.

Output Format:

- Print the sorted array on a single line, with elements separated by spaces.
- If the number of elements does not match **n**, print **-1** and do not perform any sorting.

Constraints:

- The number of elements n is between 1 and 1000.
- Each integer in the array is between -1,000,000 and 1,000,000.

Source Code:

CTC38981.c

```
#include <stdio.h>
int main(){
   int n,i,j,elements;
   scanf("%d",&n);
   int arr[n];
   for(i=0;;i++){
      scanf("%d",&arr[i]);
      elements++;
      char c;
      scanf("%c",&c);
      if(c=='\n'){
         break;
      }
   }
   for(i=1;i<n;i++){
      int key=arr[i];
      j=i-1;
      while(key<arr[j]&& j>=0){
         arr[j+1]=arr[j];
         j--;
      }
      arr[j+1]=key;
   }
   if(n==elements){
      for(i=0;i<n;i++){
         if(i<n-1){
            printf("%d ",arr[i]);
         }
         else{
            printf("%d\n",arr[i]);
         }
```

```
        }
    }
    else{
        printf("-1");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
6
10 5 3 8 6 7
3 5 6 7 8 10

Test Case - 2
User Output
3
2 4 6 8
-1

ID: 0231CYS014

S.No: 15	Exp. Name: ***Shell Sort Algorithm***	Date:2024-09-04

ID: 0231CYS014

Aim:

You are given a task to sort an array of integers using the Shell Sort algorithm. Your program should follow these specifications:

Input Format:

- The first line contains a single integer **n**, which represents the number of elements in the array.
- The second line contains **n** space-separated integers, which are the elements of the array.

Output Format:

- Print the sorted array on a single line, with elements separated by spaces.
- If the number of elements does not match **n**, print **-1** and do not perform any sorting.

Constraints:

- The number of elements n is between 1 and 1000.
- Each integer in the array is between -1,000,000 and 1,000,000.

Source Code:

CTC38982.c

```
#include <stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int arr[n],i,j,elements=0;
   for(i=0;;i++){
      scanf("%d",&arr[i]);
      elements++;
      char c = getchar();
      if(c=='\n'){
         break;
      }
   }
   if(n==elements){
      for(int gap=n/2;gap>0;gap/=2){
         for(i=gap;i<n;i++){
            int temp = arr[i];
            for(j=i;j>=gap&&arr[j-gap]>temp;j-=gap){
               arr[j]=arr[j-gap];
            }
            arr[j]=temp;
         }
      }
      for(i=0;i<n;i++){
         if(i<n-1){
            printf("%d ",arr[i]);
         }
         else{
            printf("%d",arr[i]);
         }
      }
```

```
    }
    else{
       printf("-1");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
6
10 5 3 8 6 7
3 5 6 7 8 10

Test Case - 2
User Output
3
2 4 6 8
-1

ID: 0231CYS014

S.No: 16	Exp. Name: ***Counting Sort Algorithm***	Date:2024-08-29

Aim:

You are given a task to sort an array of integers using the Counting Sort algorithm. Your program should follow these specifications:

Input Format:

- The first line contains a single integer **n**, which represents the number of elements in the array.
- The second line contains **n** space-separated integers, which are the elements of the array.

Output Format:

- Print the sorted array on a single line, with elements separated by spaces.
- If the number of elements does not match **n**, print **-1** and do not perform any sorting.

Constraints:

- The number of elements n is between 1 and 1000.
- Each integer in the array is between -1,000,000 and 1,000,000.

Source Code:

CTC38983.c

```
#include <stdio.h>
int main(){
   int n,elements=0,i,j;
   scanf("%d",&n);
   int arr[n];
   for(i=0;;i++){
      scanf("%d",&arr[i]);
      elements++;
      char c;
      c = getchar();
      if(c=='\n'){
         break;
      }
   }
   for(i=1;i<n;i++){
      int key = arr[i];
      j=i-1;
      while(key<arr[j] && j>=0){
         arr[j+1]=arr[j];
         j--;
      }
      arr[j+1]=key;
   }
   if(n==elements){
      for(i=0;i<n;i++){
         if(i<n-1){
            printf("%d ",arr[i]);
         }
         else{
            printf("%d\n",arr[i]);
         }
```

```
        }
    }
    else{
        printf("-1");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
6
10 5 3 8 6 7
3 5 6 7 8 10

Test Case - 2
User Output
3
2 4 6 8
-1

ID: 0231CYS014