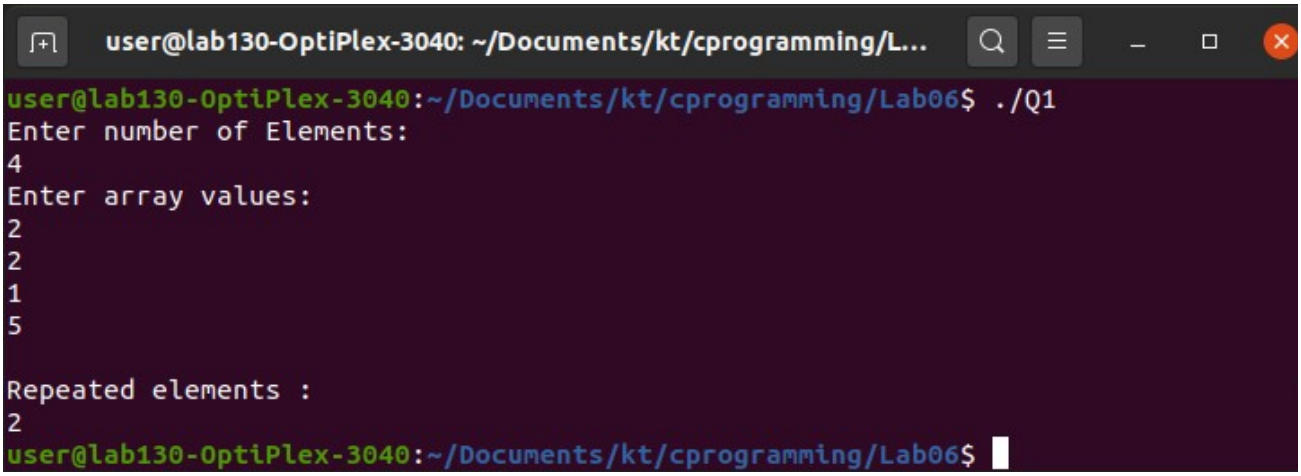
Lab06
Question 1:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int *ptr, n, i, sum = 0;
   printf("Enter number of Elements: \n");
   scanf("\n%d", &n);

   ptr=(int*)malloc(n*sizeof(int));
   if (ptr == NULL)
   {
      printf("Error!\n");
      return 0;
   }
   printf("Enter array values: \n");
   for (int i = 0; i < n; ++i)
   {
      scanf("%d", ptr+i);
      sum = sum + *(ptr + i);
   }
   printf("\nRepeated elements : \n");
   for (int i = 0; i < n; ++i)
   {
      for (int j = i+1; j < n; ++j)
      {
         if (ptr[i] == ptr[j])
         {
            printf("%d\n", ptr[j]);
         }
      }
   }
   free(ptr);
   return 0;
}
```

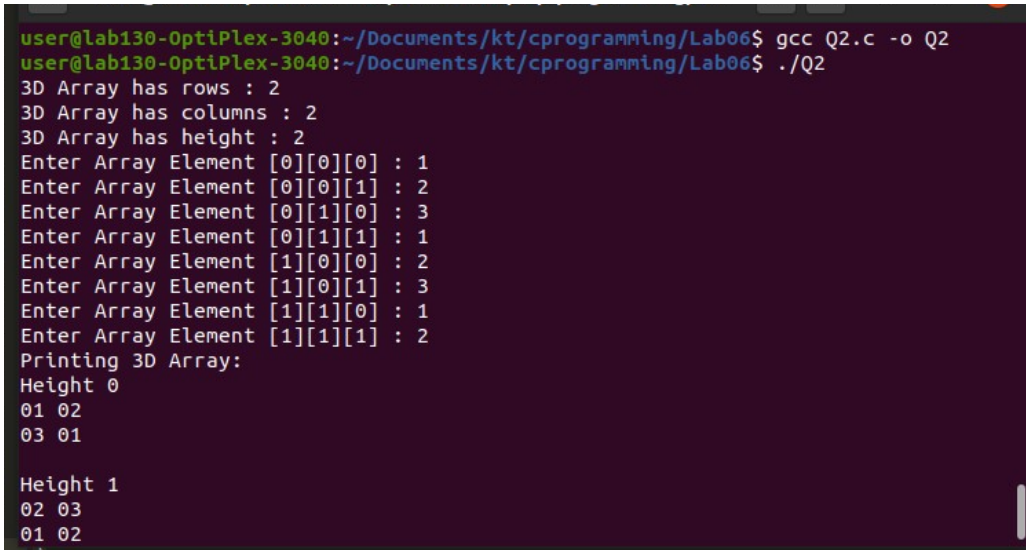
Output:



Question 2

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char* argv[])
{
  int rows = 0;
  int cols = 0;
  int height = 0;
  int ***array;
  int r, c, h;
  printf ("3D Array has rows : ");
  scanf ("%d", &rows);
  printf ("3D Array has columns : ");
  scanf ("%d", &cols);
  printf ("3D Array has height : ");
  scanf ("%d", &height);
  array = (int ***) calloc (height,sizeof(int ***));
  for (h = 0; h < height; h++) {
    array[h] = (int **) calloc(rows,sizeof(int*));
    for (r = 0; r < rows; r++) {
      array[h][r] = (int *) calloc(cols,sizeof(int));
    }
  }
  for (h = 0; h < height; h++) {
    for (r = 0; r < rows; r++) {
      for (c = 0; c < cols; c++) {
        printf ("Enter Array Element [%d][%d][%d] : ", h, r, c);
        scanf ("%d", &array[h][r][c]);
      }
    }
  }
  printf("Printing 3D Array:\n");
  for (h = 0; h < height; h++) {
    printf("Height %d\n", h);
    for (r = 0; r < rows; r++) {
      for (c = 0; c < cols; c++) {
        printf("%.2d ", array[h][r][c]);
      }
      printf("\n");
    }
    printf("\n");
  }
  return 0;
}
```

Output:

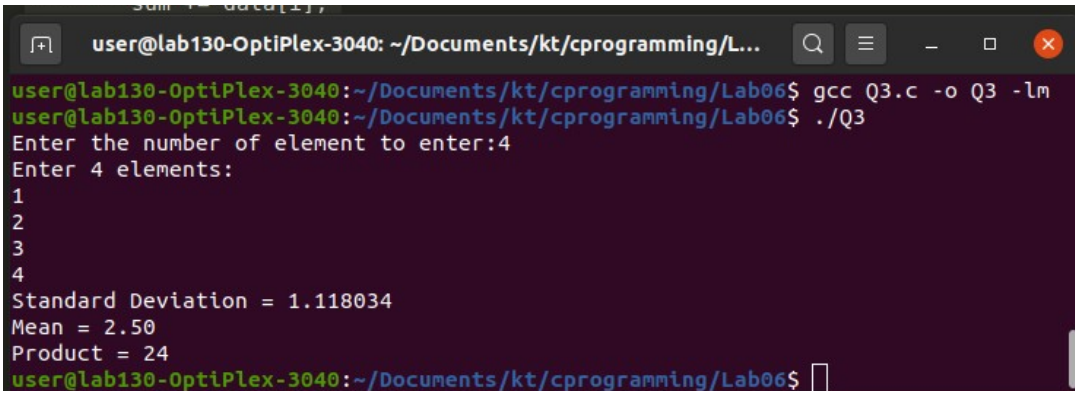


Question 3:

```
#include <math.h>
#include <stdio.h>
float sum = 0.0, mean, SD = 0.0;
int i,n;
float data[10];
//to calculate Standard deviation
float StandardDeviation(float data[]) {
   for (i = 0; i < n; ++i) {
      sum += data[i];
   }
   mean = sum / n;
   for (i = 0; i < n; ++i)
      SD += pow(data[i] - mean, 2);
   return sqrt(SD / n);
}
//print mean
int Mean(){
   printf("Mean = %.2f\n", mean);
}
//to calculate product
int product(){
   int result = 1;
   for (int i = 0; i < n; i++)
      result = result * data[i];
   printf("Product = %d \n",result );
}
//main function to call all the function
int main() {
   int i;
   printf("Enter the number of element to enter:");
   scanf("%d",&n);
   printf("Enter %d elements: \n",n);
```

```
    for (i = 0; i < n; ++i)
        scanf("%f", &data[i]);
    printf("Standard Deviation = %.6f \n", StandardDeviation(data));
    Mean();
    product();
    return 0;
}
```

Output:

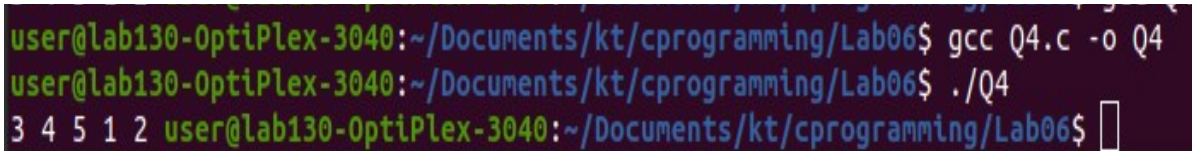


Question 3:

```
#include <stdio.h>
 /* Function to left Rotate arr[] of size n by 1*/
void leftRotatebyOne(int arr[], int n);
 /*Function to left rotate arr[] of size n by d*/
void leftRotate(int arr[], int d, int n)
{
    int i;
    for (i = 0; i < d; i++)
        leftRotatebyOne(arr, n);
}
 void leftRotatebyOne(int arr[], int n)
{
    int temp = arr[0], i;
    for (i = 0; i < n - 1; i++)
        arr[i] = arr[i + 1];
    arr[i] = temp;
}
 /* utility function to print an array */
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}
 /* Driver program to test above functions */
int main()
{
    int arr[] = { 1, 2, 3, 4, 5};
    leftRotate(arr, 2, 5);
```

```
    printArray(arr,5);
    return 0;
}
```

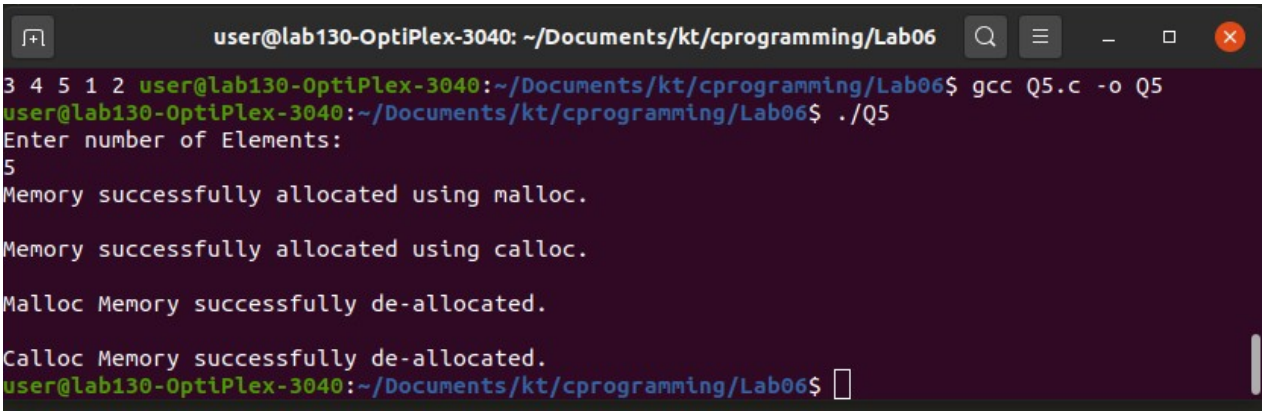
Outcome:



Question 5:

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
     // This pointer will hold the
    // base address of the block created
    int *ptr, *ptr1;
    int n, i;
    // Get the number of elements for the array
    printf("Enter number of Elements: \n");
    scanf("\n%d", &n);
    // Dynamically allocate memory using malloc()
    ptr = (int*)malloc(n * sizeof(int));

    // Dynamically allocate memory using calloc()
    ptr1 = (int*)calloc(n, sizeof(int));
    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL || ptr1 == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using malloc.\n");


        // Memory has been successfully allocated
        printf("\nMemory successfully allocated using calloc.\n");
        // Free the memory
        free(ptr);
        printf("\nMalloc Memory successfully de-allocated.\n");

        free(ptr1);
        printf("\nCalloc Memory successfully de-allocated.\n");
    }

    return 0;
}
```

Output:


user@lab130-OptiPlex-3040: ~/Documents/kt/cprogramming/Lab06
3 4 5 1 2 user@lab130-OptiPlex-3040:~/Documents/kt/cprogramming/Lab06$ gcc Q5.c -o Q5
user@lab130-OptiPlex-3040:~/Documents/kt/cprogramming/Lab06$ ./Q5
Enter number of Elements:
5
Memory successfully allocated using malloc.
Memory successfully allocated using calloc.
Malloc Memory successfully de-allocated.
Calloc Memory successfully de-allocated.
user@lab130-OptiPlex-3040:~/Documents/kt/cprogramming/Lab06$