```c
/*
You have an array A of N elements.
Create all possible ordered triplets (i,j,k) such that I,j,k are pairwise
distinct and 1<=i and j, k <=N.
The value of this triplet will be (Ai-Aj)*Ak. Now, find out the maximum value
among all possible ordered triplets.
Note: Two ordered triplets (a,b,c) and (d,e,f) are only equal when a=d, b=e,
c=f. (1,2,3) and (2,1,3)
are two different ordered triplets.
ARKAPRATIM GHOSH 13000121058
*/

#include<stdio.h>
#include<stdlib.h>
void merge(int *p,int l,int m,int r);
void merge_sort(int *p,int l,int r);
void  main()
{
        int i,*p,n,ch;
        int a,b,c,in=-1;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    fflush(stdin);
    p=(int *)malloc(n*sizeof(int));
    for(i=0;i<n;i++)
    {
        printf("Enter the element %d:",(i+1));
        scanf("%d",&p[i]);
        fflush(stdin);
    }
    printf("The Array is: ");
    for(i=0;i<n;i++)
    {
        printf("%d ",p[i]);
    }
    merge_sort(p,0,n-1);

    in=n-1;
    a=p[n-1];
    //printf("%d",a);
    for(i=0;i<n-1;i++)
    {
            if(p[i]==a)
            {
                    in=i;
                    break;
            }
    }
    b=p[in-1];
    //in=n-2;
    in=n-2;
    for(i=0;i<n-1;i++)
    {
            if(p[i]==b)
            {
                    in=i;
                    break;
            }
    }
    c=p[in-1];
    printf("\nThe triplet is %d %d %d\n",c,b,a);
    printf("\nThe maximum value is %d\n",((c-b)*a));

}
void merge(int *p,int l,int m,int r)
{
    int i,j,k;
    int n1=m-l+1;
    int n2=r-m;
    int *L,*R;
    L=(int *)malloc(n1*sizeof(int));
    R=(int *)malloc(n2*sizeof(int));
    for(i=0;i<n1;i++)
    {
         L[i]=p[l+i];
    }
    for(j=0;j<n2;j++)
    {
        R[j]=p[m+1+j];
    }
    i=0;j=0;k=l;
    while(i<n1 && j<n2)
    {
        if(L[i]<R[j])
        {
            p[k]=L[i];
            i++;
        }
        else
        {
            p[k]=R[j];
            j++;
        }
        k++;
    }
    while(i<n1)
    {
        p[k]=L[i];
        i++;
        k++;
        }
        while(j<n2)
        {
            p[k]=R[j];
            j++;
            k++;
        }
    }
    void merge_sort(int *p,int l,int r)
    {
         int m;
         if(l<r)
         {
            m=l+(r-l)/2;
            merge_sort(p,l,m);
            merge_sort(p,m+1,r);
            merge(p,l,m,r);
         }
    }
```