BATCHERS' SORT

```
#include <stdio.h>

#include <stdlib.h>

int minimum(int x, int y)

{

   if (x < y)

   {

      return x;

   }

   else

   {

      return y;

   }

}


int maximum(int x, int y)

{

   if (x > y)

   {

      return x;

   }

   else

   {

      return y;

   }

}


int *batcher(int *U, int *V, int m, int n)

{

   int *S;

   if (m == 0 && n == 0)
```

```
{
    return NULL;
}
else if (n == 0)
{
    S = (int *)malloc(m * sizeof(int));
    for (int i = 0; i < m; i++)
    {
        S[i] = U[i];
    }
}
else if (m == 0)
{
    S = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
    {
        S[i] = V[i];
    }
}
else if (m == 1 && n == 1)
{
    S = (int *)malloc(2 * sizeof(int));
    S[0] = minimum(U[0], V[0]);
    S[1] = maximum(U[0], V[0]);
}
else
{
    int *Ou = (int *)malloc(((m + 1) / 2) * sizeof(int));
    int k = 0;
    for (int i = 0; i < m; i = i + 2)
    {
```

```
    Ou[k++] = U[i];
}
int *Ov = (int *)malloc(((n + 1) / 2) * sizeof(int));
k = 0;
for (int i = 0; i < n; i = i + 2)
{
    Ov[k++] = V[i];
}
int *A = batcher(Ou, Ov, (m + 1) / 2, (n + 1) / 2);
free(Ou);
free(Ov);

int *Eu = (int *)malloc((m / 2) * sizeof(int));
k = 0;
for (int i = 1; i < m; i = i + 2)
{
    Eu[k++] = U[i];
}
int *Ev = (int *)malloc((n / 2) * sizeof(int));
k = 0;
for (int i = 1; i < n; i = i + 2)
{
    Ev[k++] = V[i];
}
int *B = batcher(Eu, Ev, (m / 2), (n / 2));
free(Eu);
free(Ev);

int c;
if ((m % 2 == 0) && (n % 2 == 0))
{
```

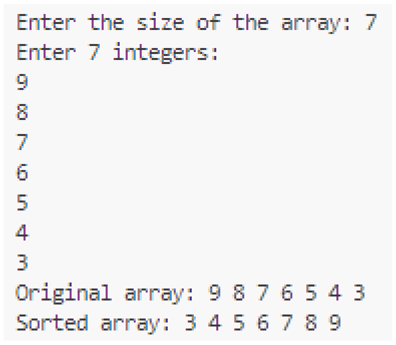
```
            c = ((m / 2) + (n / 2)) - 1;
        }
        else
        {
            c = (m / 2) + (n / 2);
        }
        k = 0;
        S = (int *)malloc((m + n) * sizeof(int));
        S[k++] = A[0];
        for (int i = 1; i <= c; i++)
        {
            S[k++] = minimum(A[i], B[i - 1]);
            S[k++] = maximum(A[i], B[i - 1]);
        }
        if ((m % 2 == 0) && (n % 2 == 0))
        {
            S[k++] = B[(m / 2) + (n / 2) - 1];
        }
        else if ((m % 2 != 0) && (n % 2 != 0))
        {
            S[k++] = A[(m / 2) + (n / 2) + 1];
        }
        free(A);
        free(B);
    }
    return S;
}

int *batcher_sort(int arr[], int start, int no)
{
    int *S;
    if (no > 1)
    {
        int *U = batcher_sort(arr, start, no / 2);
        int *V = batcher_sort(arr, start + no / 2, no - no / 2);
        S = batcher(U, V, no / 2, no - no / 2);
        free(U);
        free(V);
    }
    else
    {
        S = (int *)malloc(sizeof(int));
        S[0] = arr[start];
    }
    return S;
}

int main()
{
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers: \n", n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Original array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
```

```c
    }
    printf("\n");
    int *array = batcher_sort(arr, 0, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", array[i]);
    }
    printf("\n");
    free(array);
    return 0;
}
```

OUTPUT



```
Enter the size of the array: 7
Enter 7 integers:
9
8
7
6
5
4
3
Original array: 9 8 7 6 5 4 3
Sorted array: 3 4 5 6 7 8 9
```