Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

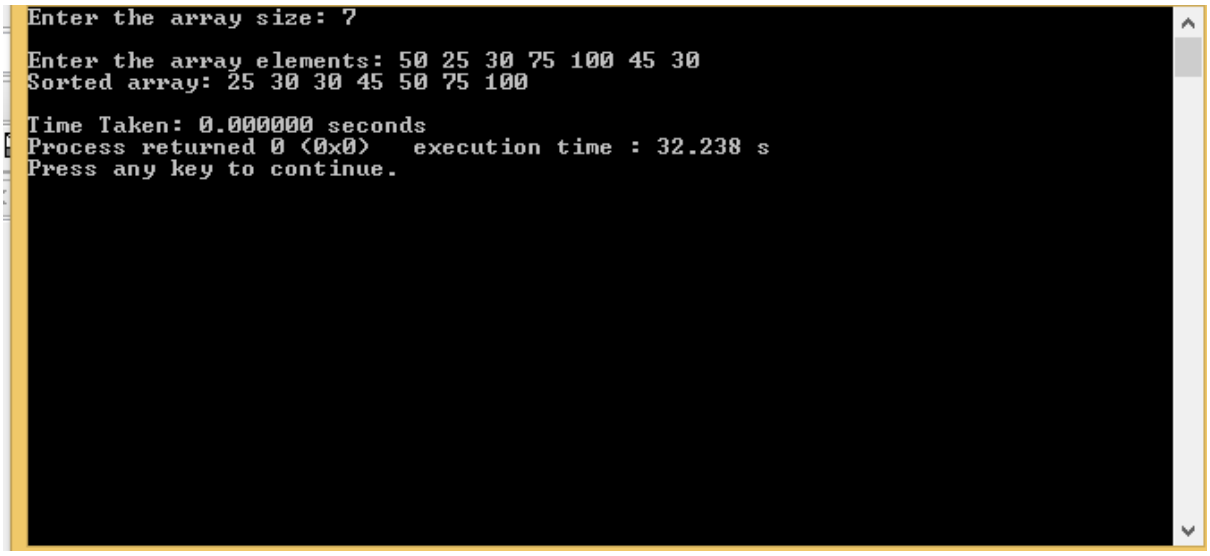
CODE:

```
#include <stdio.h>
#include <time.h>
#include<stdlib.h>

void swap(long* a, long* b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}

void heapify(long arr[], int n, int i) {
   int largest = i;
   int left = 2 * i + 1;
   int right = 2 * i + 2;

   if (left < n && arr[left] > arr[largest])
      largest = left;

   if (right < n && arr[right] > arr[largest])
      largest = right;

   if (largest != i) {
      swap(&arr[i], &arr[largest]);

      heapify(arr, n, largest);
   }
```

```
}

void heapSort(long arr[], int n) {

    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
}

void printArray(long arr[], int n) {
    for (int i = 0; i < n; ++i)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {

    int n ;
    long arr[100000];
    printf("Enter the array size: ");
    scanf("%d", &n);
     printf("\nEnter the array elements: ");
    for(int i = 0 ; i < n ; i++){
        scanf("%d",&arr[i]);
       // arr[i] = rand() % 100000;
```

```
    }

    clock_t start = clock();
    heapSort(arr, n);
    clock_t end = clock();
    double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

    printf("Sorted array: ");
    printArray(arr, n);
    printf("\nTime Taken: %f seconds", time_taken);
    return 0;
}
```

OUTPUT:



Implement “N-Queens Problem” using Backtracking.

CODE:

```
#include<stdio.h>
#include<math.h>

int board[20],count;

int main()
{
   int n,i,j;

   void queen(int row,int n);

   printf("N Queens Problem Using Backtracking:");
   printf("\n\nEnter number of Queens:");
   scanf("%d",&n);

   queen(1,n);
   return 0;
}

void print(int n)
{
   int i,j;
   printf("\n\nSolution %d:\n\n",++count);

   for(i=1;i<=n;++i)
      printf("\t%d",i);

   for(i=1;i<=n;++i)
   {
```

```
        printf("\n\n%d",i);
        for(j=1;j<=n;++j)
        {
            if(board[i]==j)
                printf("\tQ");

            else
                printf("\t-");
        }
    }
}

int place(int row,int column)
{
    int i;
    for(i=1;i<=row-1;++i)
    {
        if(board[i]==column)
            return 0;

        else
            if(abs(board[i]-column)==abs(i-row))
            return 0;
    }
    return 1;
}

void queen(int row,int n)
{
    int column;
    for(column=1;column<=n;++column)
```

```
  {
    if(place(row,column))
    {
      board[row]=column;
      if(row==n)
        print(n);
      else
        queen(row+1,n);
    }
  }
}
```

OUTPUT:

```
N Queens Problem Using Backtracking:
Enter number of Queens:4

Solution 1:
        1       2       3       4
1       -       Q       -       -
2       -       -       -       Q
3       Q       -       -       -
4       -       -       Q       -
Solution 2:
        1       2       3       4
1       -       -       Q       -
2       Q       -       -       -
3       -       -       -       Q
4       -       Q       -       -
Process returned 0 (0x0)   execution time : 2.757 s
Press any key to continue.
```