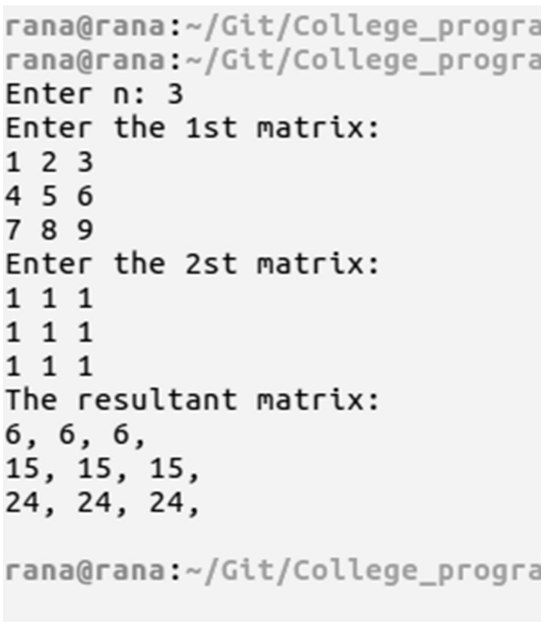
Source code:

```
#include <iostream>
#include <vector>
#include <tuple>

typedef std::vector<std::vector<int>> data;

data merge(const data &a, const data &b, const data &c, const data &d)
{
  int n = a.size()*2;
  data res(n, std::vector<int>(n,0));

  for(int i = 0, x = 0; i < n/2; i++, x++)
        for(int j = 0, y = 0; j < n/2; j++, y++)
              res[i][j] = a[x][y];

  for(int i = 0, x = 0; i < n/2; i++, x++)
        for(int j = n/2, y = 0; j < n; j++, y++)
              res[i][j] = b[x][y];

  for(int i = n/2, x = 0; i < n; i++, x++)
        for(int j = 0, y = 0; j < n/2; j++, y++)
              res[i][j] = c[x][y];

  for(int i = n/2, x = 0; i < n; i++, x++)
        for(int j = n/2, y = 0; j < n; j++, y++)
              res[i][j] = d[x][y];
  return res;
}

std::tuple<data, data, data, data> slice(const
std::vector<std::vector<int>> &mat)
{
  int n = mat.size();
  std::vector<std::vector<int>> ans1(n/2, std::vector<int>(n/2));
  std::vector<std::vector<int>> ans2(n/2, std::vector<int>(n/2));
  std::vector<std::vector<int>> ans3(n/2, std::vector<int>(n/2));
  std::vector<std::vector<int>> ans4(n/2, std::vector<int>(n/2));
  for(int i = 0; i < n/2; i++)
        for(int j = 0; j < n/2; j++)
              ans1[i][j] = mat[i][j];

  for(int i = 0, x = 0; i < n/2; i++, x++)
        for(int j = n/2, y = 0; j < n; j++, y++)
              ans2[x][y] = mat[i][j];

  for(int i = n/2, x = 0; i < n; i++, x++)
        for(int j = 0, y = 0; j < n/2; j++, y++)
              ans3[x][y] = mat[i][j];

  for(int i = n/2, x = 0; i < n; i++, x++)
        for(int j = n/2, y = 0; j < n; j++, y++)
              ans4[x][y] = mat[i][j];
  return std::make_tuple(ans1, ans2, ans3, ans4);
}

data operator-(const data& a, const data& b)
{
  std::vector<std::vector<int>> c(a.size(), std::vector<int>(a.size()));
  for(int i=0;i<a.size();i++)
        for(int j=0;j<a.size();j++)
```

```
            c[i][j]=a[i][j]-b[i][j];
    return c;
}

data operator+(const data& a, const data& b)
{
    std::vector<std::vector<int>> c(a.size(), std::vector<int>(a.size()));
    for(int i=0;i<a.size();i++)
        for(int j=0;j<a.size();j++)
            c[i][j]=a[i][j]+b[i][j];
    return c;
}

data product(data X, data Y)
{
    int n = X.size();
    if(n==2)
    {
        data bound(2,std::vector<int>(2));
        bound[0][0] = (X[0][0]*Y[0][0])+(X[0][1]*Y[1][0]);
        bound[0][1] = (X[0][0]*Y[0][1])+(X[0][1]*Y[1][1]);
        bound[1][0] = (X[1][0]*Y[0][0])+(X[1][1]*Y[1][0]);
        bound[1][1] = (X[1][0]*Y[0][1])+(X[1][1]*Y[1][1]);
        return bound;
    }
    if(n%2==1)
    {
        for(int i=0;i<X.size();i++)
        {
            X[i].push_back(0);
            Y[i].push_back(0);
        }
        X.push_back(std::vector<int>(X.size(),0));
        Y.push_back(std::vector<int>(Y.size(),0));
        Y[Y.size()-1][Y.size()-1]=1;
        X[X.size()-1][X.size()-1]=1;
    }
    data A, B, C, D, E, F, G, H;
    std::tie(A, B, C, D) = slice(X);
    std::tie(E, F, G, H) = slice(Y);
    data P1 = product(A, F-H);
    data P2 = product(A+B, H);
    data P3 = product(C+D, E);
    data P4 = product(D, G-E);
    data P5 = product(A+D, E+H);
    data P6 = product(B-D, G+H);
    data P7 = product(A-C, E+F);
    auto temp = merge((P6+P5)+(P4-P2), P1+P2, P3+P4, (P1+P5)-(P3+P7));
    if(n%2==1)
    {
        temp.pop_back();
        for(auto& i: temp)
            i.pop_back();
    }
    return temp;
}

int main()
{
    int n;
    std::cout<<"Enter n: ";
    std::cin>>n;
    data mtx1(n, std::vector<int>(n)), mtx2(n, std::vector<int>(n));
    std::cout<<"Enter the 1st matrix:\n";
    for(auto& i : mtx1)
          for(auto& j : i)
                std::cin>>j;
    std::cout<<"Enter the 2st matrix:\n";
    for(auto& i : mtx2)
          for(auto& j : i)
                std::cin>>j;
    std::cout<<"The resultant matrix:\n";
    auto res = product(mtx1,mtx2);
    for(auto& i : res)
    {
          for(auto& j : i)
                std::cout<<j<<", ";
          std::cout<<"\n";
    }
    std::cout<<std::endl;
    return 0;
}
```

Screen-Shot:



```
rana@rana:~/Git/College_progra
rana@rana:~/Git/College_progra
Enter n: 3
Enter the 1st matrix:
1 2 3
4 5 6
7 8 9
Enter the 2st matrix:
1 1 1
1 1 1
1 1 1
The resultant matrix:
6, 6, 6,
15, 15, 15,
24, 24, 24,

rana@rana:~/Git/College_progra
```

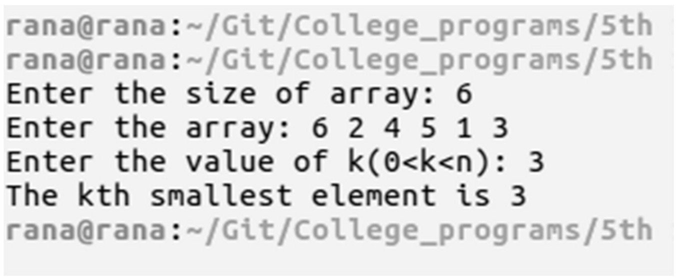
Time Complexity:

Source Code:

```
#include <iostream>
#include <vector>

inline void swap(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}

void heapify(std::vector<int> &heap, int pos)
{
  int large=heap[pos], i = pos;
  int left = (2*pos)+1;
  int right = (2*pos)+2;
  if(left<heap.size())
        if(large<heap[left])
        {
              large = heap[left];
              i = left;
        }
  if(right<heap.size())
        if(large<heap[right])
        {
              large = heap[right];
              i = right;
        }
  if(pos!=i)
  {
        swap(heap[pos], heap[i]);
        heapify(heap, i);
  }
}

int kth_small(std::vector<int> &vect, int k)
{
  int n = vect.size();
  std::vector<int> kheap(vect.begin(), vect.begin()+(k+1));
  for(int i=(k-1)/2;i>=0;i--)
        heapify(kheap, i);
  for(int i=k+1;i<n;i++)
        if(kheap[0]>vect[i])
        {
              swap(kheap[0], vect[i]);
              heapify(kheap, 0);
        }
  return kheap[0];
}

int main()
{
  int n, k;
  std::cout<<"Enter the size of array: ";
  std::cin>>n;
  std::vector<int> arr(n);
  std::cout<<"Enter the array: ";
  for(auto &i: arr)
        std::cin>>i;
  std::cout<<"Enter the value of k(0<k<n): ";
  std::cin>>k;
```

```
   std::cout<<"The kth smallest element is "<<kth_small(arr, k-
1)<<std::endl;
}
```

Screen-shot:



```
rana@rana:~/Git/College_programs/5th
rana@rana:~/Git/College_programs/5th
Enter the size of array: 6
Enter the array: 6 2 4 5 1 3
Enter the value of k(0<k<n): 3
The kth smallest element is 3
rana@rana:~/Git/College_programs/5th
```

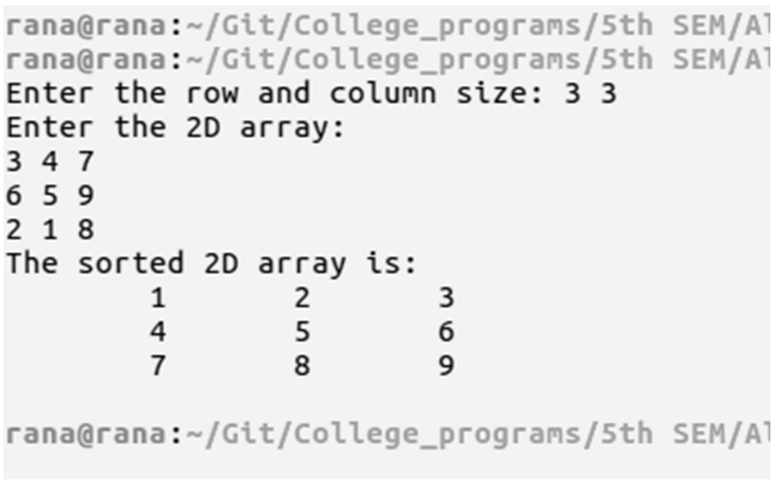
Time Complexity:

Source Code:

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

void qsrt(int left, int right, int **arr, int n, int m)
{
  if(left>=right-2)
        return;
  int pivot = arr[(left+1)/m][(left+1)%m], i=0, j, temp, end = right,
                                                         start=left;
  while(left!=right-1)
  {
        if(i%2 == 0)
        {
              if(pivot>=arr[(left+1)/m][(left+1)%m])
                    left++;
              else
              {
                    temp = arr[(left+1)/m][(left+1)%m];
                    arr[(left+1)/m][(left+1)%m] = arr[(right-
                                                  1)/m][(right-1)%m];
                    arr[(right-1)/m][(right-1)%m] = temp;
                    right--;
              }
        }
        else
        {
              if(pivot<=arr[(right-1)/m][(right-1)%m])
                    right--;
              else
              {
                    temp = arr[(left+1)/m][(left+1)%m];
                    arr[(left+1)/m][(left+1)%m] = arr[(right-
                                                  1)/m][(right-1)%m];
                    arr[(right-1)/m][(right-1)%m] = temp;
                    left++;
              }
        }
        i++;
  }
  if(left!=start)
  {
        arr[(start+1)/m][(start+1)%m] = arr[left/m][left%m];
        arr[left/m][left%m] = pivot;
  }
  qsrt(start, left, arr, n, m);
  qsrt(left, end, arr, n, m);
}

int main()
{
  int **arr, i, j, n, m;
  printf("Enter the row and column size: ");
  scanf("%d%d",&n, &m);
  arr = (int **)malloc(n*sizeof(int *));
  for(i=0;i<n;i++)
        arr[i] = (int *)malloc(m*sizeof(int));
  printf("Enter the 2D array:\n");
  for(i=0;i<n;i++)
        for(j=0;j<m;j++)
              scanf("%d", &arr[i][j]);
```

```
    qsrt(-1, n*m, arr, n, m);
    printf("The sorted 2D array is:\n");
    for(i=0;i<n;i++)
    {
          for(j=0;j<m;j++)
                printf("\t%d", arr[i][j]);
          printf("\n");
    }
    printf("\n");
    return 0;
  }
```

Screen-shot:



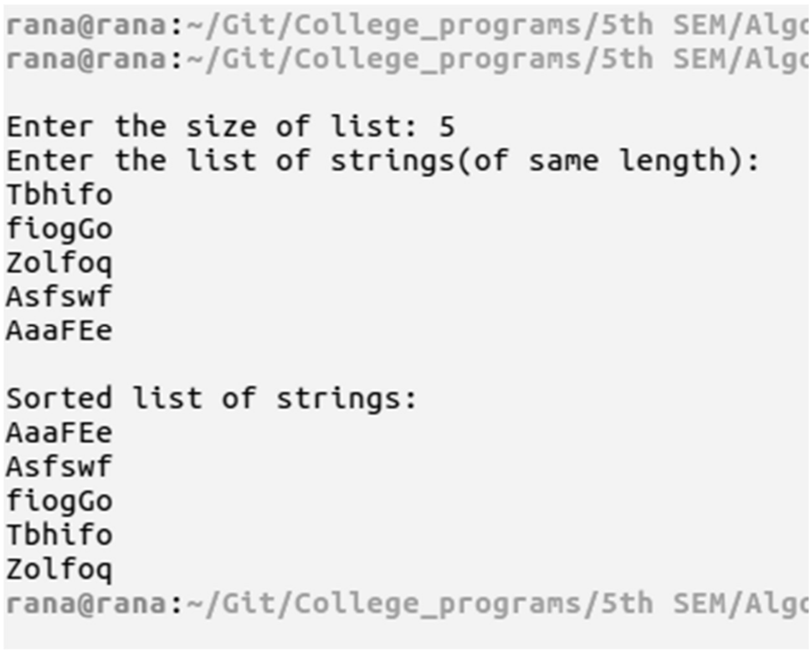
```
rana@rana:~/Git/College_programs/5th SEM/Al
rana@rana:~/Git/College_programs/5th SEM/Al
Enter the row and column size: 3 3
Enter the 2D array:
3 4 7
6 5 9
2 1 8
The sorted 2D array is:
        1       2       3
        4       5       6
        7       8       9

rana@rana:~/Git/College_programs/5th SEM/Al
```

Time Complexity:

Source Code:

```
#include <iostream>
#include <vector>
#include <tuple>

inline void upper_case(std::tuple<std::string,std::string> &str)
{
  int size = std::get<1>(str).size();
  for(int i=0;i<size;i++)
        std::get<0>(str).push_back(tolower(std::get<1>(str)[i]));
}

void radixsort(std::vector<std::tuple<std::string,std::string>> &list)
{
  int len = std::get<0>(list[0]).size(), n = list.size();
  std::vector<std::tuple<std::string,std::string>> temp(n);
  for(int i=len-1;i>=0;i--)
  {
        int count[26]={0};
        for(auto &j: list)
              count[std::get<0>(j)[i]-'a']++;
        for(int j=1;j<26;j++)
              count[j]+=count[j-1];
        for(int j=n-1;j>=0;j--)
        {
              temp[count[std::get<0>(list[j])[i]-'a']-1] = list[j];
              count[std::get<0>(list[j])[i]-'a']--;
        }
        for(int j=0;j<n;j++)
              list[j]=temp[j];
  }
}

int main()
{
  int n, i;
  std::cout<<"\nEnter the size of list: ";
  std::cin>>n;
  std::vector<std::tuple<std::string, std::string>> list(n);
  std::cout<<"Enter the list of strings(of same length):\n";
  for(auto &i: list)
  {
        std::cin>>std::get<1>(i);
        upper_case(i);
  }
  radixsort(list);
  std::cout<<"\nSorted list of strings:\n";
  for(auto &i: list)
        std::cout<<std::get<1>(i)<<std::endl;
}
```

Screen-shot:



```
rana@rana:~/Git/College_programs/5th SEM/Algo
rana@rana:~/Git/College_programs/5th SEM/Algo

Enter the size of list: 5
Enter the list of strings(of same length):
Tbhifo
fiogGo
Zolfoq
Asfswf
AaaFEe

Sorted list of strings:
AaaFEe
Asfswf
fiogGo
Tbhifo
Zolfoq
rana@rana:~/Git/College_programs/5th SEM/Algo
```

Time Complexity: