LAB ASSESSMENT 1

Name: Karan Sehgal

Registration Number: 22BCE3939

Course Name: Design and Analysis of Algorithms

Course Code: BCSE204P

Submitted to: Prof. Gayatri P

Question 1:

Fractional Knapsack problem using greedy Approach.

Problem Statement:

Given the weights W and profits P of N items which are to be inserted in the knapsack which can bear the weight up to M. Insert maximum value in the knapsack with weight not greater than K. Every item can be inserted in fractional manner also (that is we can break the item to maximize the value in the knapsack).

Pseudo-Code:

```
Fractional knapsack problem using greedy Algorithm
22BCE3939
function knapsack (n, weight[], profit[], cap)
 initialize x array of size n with all
  elements = 0
 initialize tp to 0
 set u to cap
 for i from 0 to n-1
       set x[i] to 0.0
 for i from 0 to n-1
     if weight[i] > u
        break
     else
        set x[i] to 1.0
        add profit[i] to tp
        subtract weight[i] from u
 if i<n
      set x[i] to u/ weight[i]
 add x[i] * profit[i] to tp
```

22BCE3939

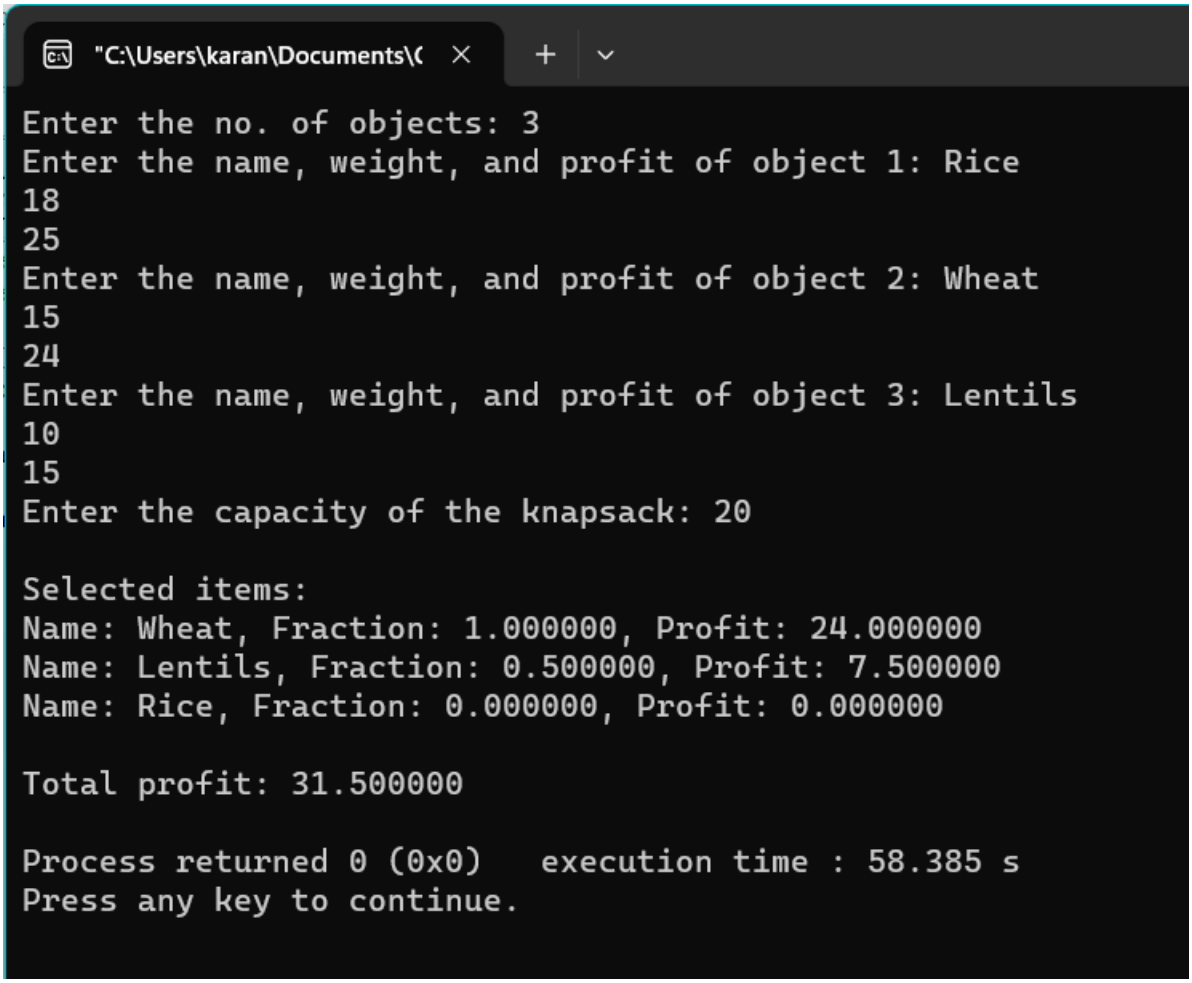
```
function main()
    initialize weight array of size 20
    initialize profit array of size 20
    initialize ratio array of size 20
    initialize temp and temp1 variables
    initialize name array of size 20X20

    read n from user
    for i from 0 to n-1
        read name[i], weight[i], profit[i]
                from user
    read capacity from user
    for i from 0 to n-1
        set ratio[i] to profit[i]/weight[i]
    for i from 0 to n-1        // sorting all arrays
        for j from i+1 to n-1  according to
            if (ratio[i] < ratio[j])   ratio[]
                swap ratio[i] with ratio[j]
// using temp & temp1
                swap weight[i] with weight[j]
                swap profit[i] with profit[j]
                swap name[i] with name[j]
    call Knapsack(n, weight, profit, capacity)
    print "selected items".
    for i from 0 to n-1
        print "Name: Fraction: Profit:",
                name[i], x[i], profit[i] * x[i]
    print "Total Optimal Profit", tp
```

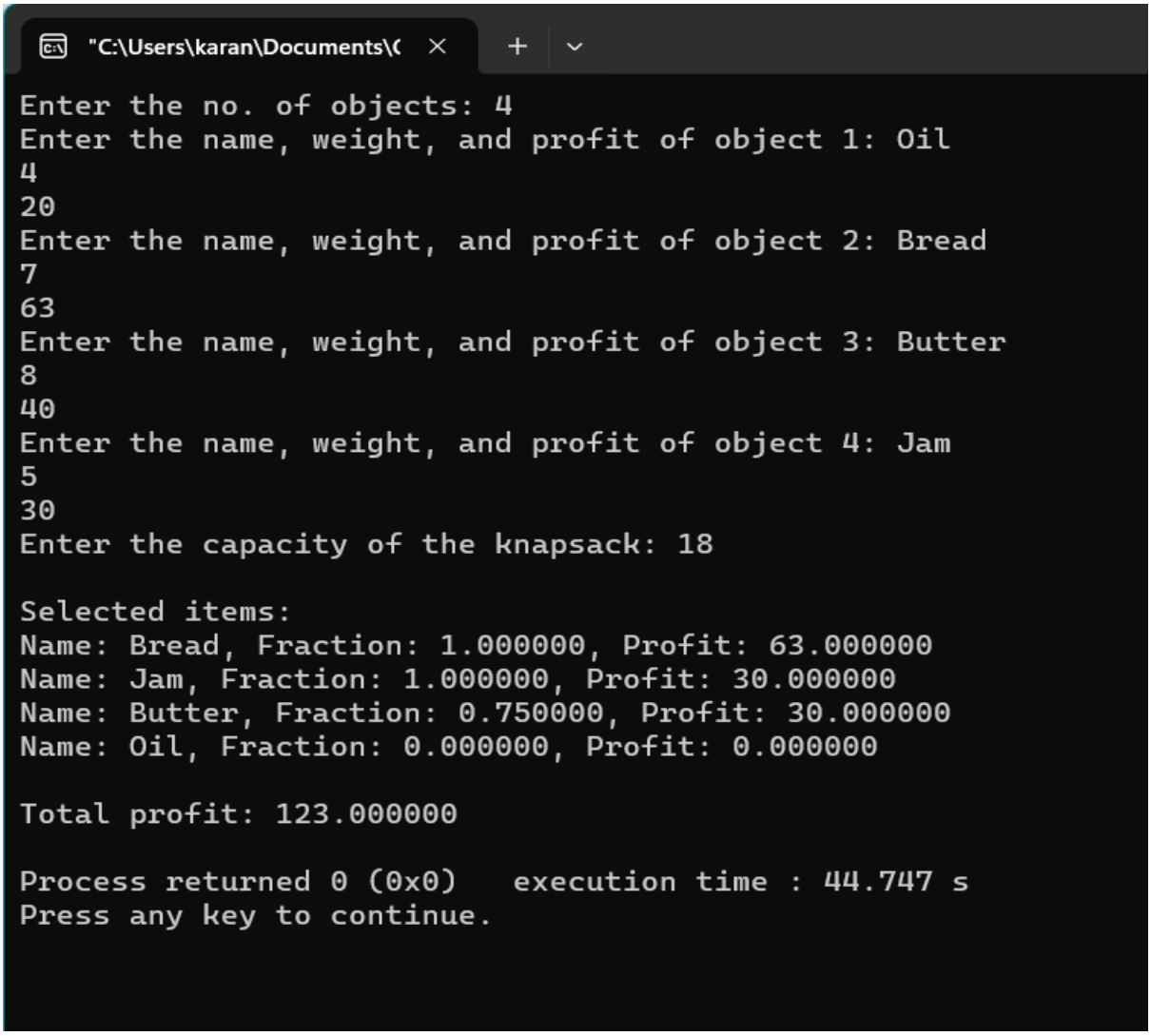
Source code:

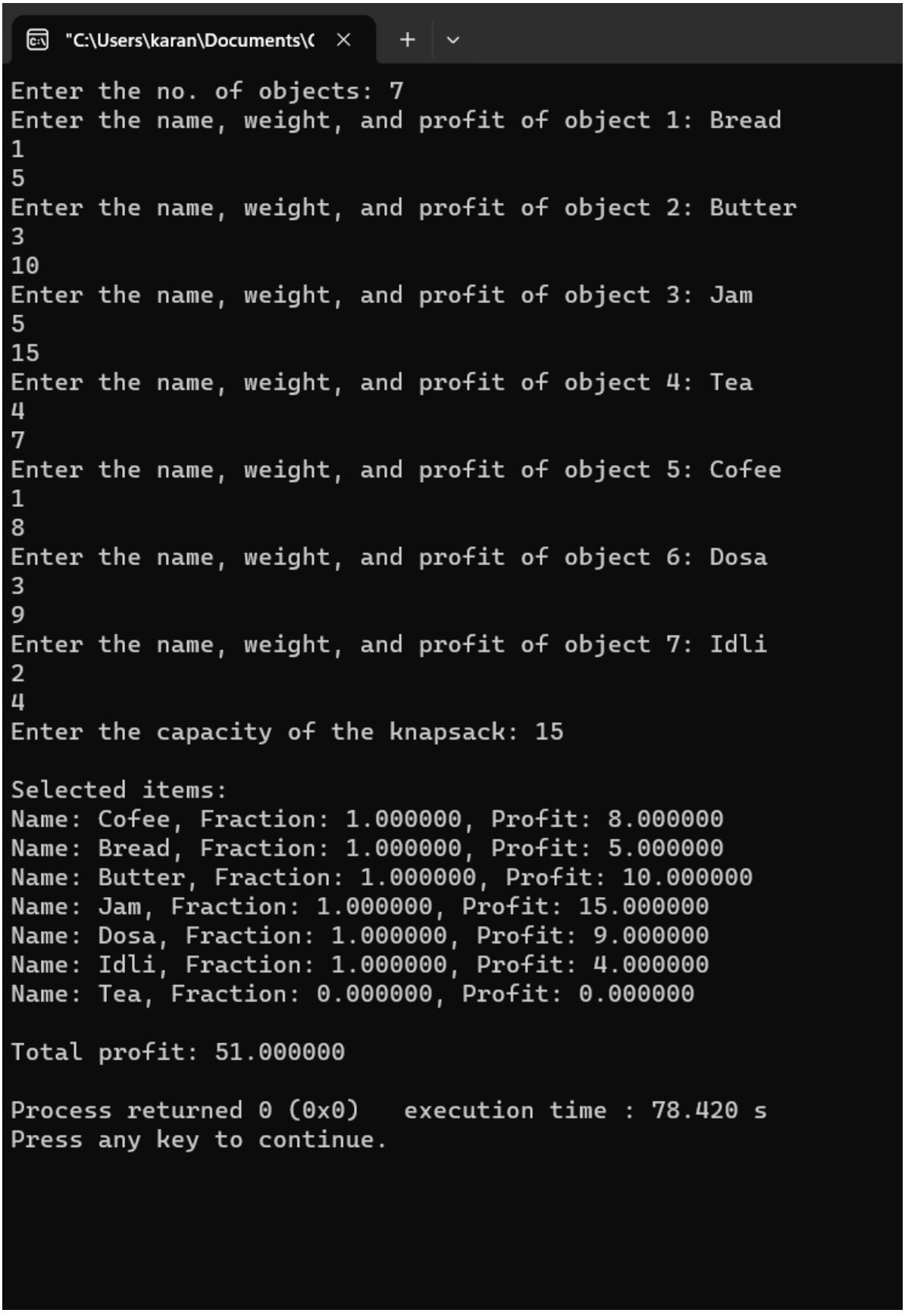
```
#include<stdio.h>
#include<string.h>

float x[20];
float tp = 0;

void knapsack(int n, float weight[], float profit[], float capacity) {
    int i, j;
    float u;

    u = capacity;

    for (i = 0; i < n; i++)
        x[i] = 0.0;

    for (i = 0; i < n; i++) {
        if (weight[i] > u)
            break;
        else {
            x[i] = 1.0;
            tp = tp + profit[i];
            u = u - weight[i];
        }
    }

    if (i < n)
        x[i] = u / weight[i];

    tp = tp + (x[i] * profit[i]);
}

int main() {
    float weight[20], profit[20], capacity;
    int n, i, j;
    float ratio[20], temp;
    char temp1[20];
    char name[20][20];

    printf("Enter the no. of objects: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter the name, weight, and profit of object %d: ", i + 1);
```

```
        scanf("%s", name[i]);
        scanf("%f", &weight[i]);
        scanf("%f", &profit[i]);
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%f", &capacity);

    for (i = 0; i < n; i++) {
        ratio[i] = profit[i] / weight[i];
    }

    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (ratio[i] < ratio[j]) {
                temp = ratio[j];
                ratio[j] = ratio[i];
                ratio[i] = temp;

                temp = weight[j];
                weight[j] = weight[i];
                weight[i] = temp;

                temp = profit[j];
                profit[j] = profit[i];
                profit[i] = temp;

                strcpy(temp1, name[j]);
                strcpy(name[j], name[i]);
                strcpy(name[i], temp1);
            }
        }
    }

    knapsack(n, weight, profit, capacity);

    printf("\nSelected items:\n");
    for (int i = 0; i < n; i++) {
        printf("Name: %s, Fraction: %f, Profit: %f\n", name[i], x[i], profit[i] *
x[i]);
    }

    printf("\nTotal profit: %f\n", tp);

    return 0;}
```

Output :


"C:\Users\karan\Documents\(
Enter the no. of objects: 3
Enter the name, weight, and profit of object 1: Rice
18
25
Enter the name, weight, and profit of object 2: Wheat
15
24
Enter the name, weight, and profit of object 3: Lentils
10
15
Enter the capacity of the knapsack: 20
Selected items:
Name: Wheat, Fraction: 1.000000, Profit: 24.000000
Name: Lentils, Fraction: 0.500000, Profit: 7.500000
Name: Rice, Fraction: 0.000000, Profit: 0.000000
Total profit: 31.500000
Process returned 0 (0x0) execution time : 58.385 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter the no. of objects: 4
Enter the name, weight, and profit of object 1: Oil
4
20
Enter the name, weight, and profit of object 2: Bread
7
63
Enter the name, weight, and profit of object 3: Butter
8
40
Enter the name, weight, and profit of object 4: Jam
5
30
Enter the capacity of the knapsack: 18
Selected items:
Name: Bread, Fraction: 1.000000, Profit: 63.000000
Name: Jam, Fraction: 1.000000, Profit: 30.000000
Name: Butter, Fraction: 0.750000, Profit: 30.000000
Name: Oil, Fraction: 0.000000, Profit: 0.000000
Total profit: 123.000000
Process returned 0 (0x0) execution time : 44.747 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter the no. of objects: 7
Enter the name, weight, and profit of object 1: Bread
1
5
Enter the name, weight, and profit of object 2: Butter
3
10
Enter the name, weight, and profit of object 3: Jam
5
15
Enter the name, weight, and profit of object 4: Tea
4
7
Enter the name, weight, and profit of object 5: Cofee
1
8
Enter the name, weight, and profit of object 6: Dosa
3
9
Enter the name, weight, and profit of object 7: Idli
2
4
Enter the capacity of the knapsack: 15
Selected items:
Name: Cofee, Fraction: 1.000000, Profit: 8.000000
Name: Bread, Fraction: 1.000000, Profit: 5.000000
Name: Butter, Fraction: 1.000000, Profit: 10.000000
Name: Jam, Fraction: 1.000000, Profit: 15.000000
Name: Dosa, Fraction: 1.000000, Profit: 9.000000
Name: Idli, Fraction: 1.000000, Profit: 4.000000
Name: Tea, Fraction: 0.000000, Profit: 0.000000
Total profit: 51.000000
Process returned 0 (0x0) execution time : 78.420 s
Press any key to continue.

Question 2:

Huffman Encoding using greedy Approach.

Pseudo-Code:

```
Huffman Encoding using Greedy Approach
function newnode (x, f)                    22BCE3939
allocate memory for a new node
set left & right child pointers to NULL
set node's data to x
set node's frequency to f
return newly created node.
function createminheap (cap)
temp = allocate memory for a new
       minheap
temp.size = 0
temp.cap = cap
temp.array = allocate memory for an
array of minheapnode pointers of size
 cap.
return temp
                          // standard heapify fn
function heapify (curheap, ind)
left = 2 * ind + 1
right = 2 * ind + 2
min = ind
if left < curheap.size and curheap.array-
 [left].freq < curheap.array[min].freq
 min = left
if right < curheap.size and curheap.array-
 [right].freq < curheap.array[min].freq
 min = right
```

22BCE3939

```
if min != ind
    swap minheap(curheap.array[min],
                 curheap.array[ind])

    heapify(curheap, min)

// To extract min value node from heap
function extractmin(curheap)
    temp = curheap.array[0]
    curheap.array[0] = curheap.array[curheap.size - 1]
    curheap.size = curheap.size - 1
    callheapify(curheap, 0)
    return temp

// A utility function to insert new node to heap
function insertminheap(curheap, curnode)
    curheap.size ++
    i = curheap.size - 1
    while i and curnode.freq < curheap.
        array[(i-1)/2].freq
        curheap.array[i] = curheap.array[(i-1)/2]
        i = (i-1)/2
    curheap.array[i] = curnode

// A standard fn to build min heap
function buildminheap(curheap)
    set n to curheap.size - 1
    for i from (n-1)/2 to 0, i decrement
        heapify(curheap, i)
fn printarr(arr, n)
    for i from 0 to n-1
        print arr[i]
    print new line.
```

```
function creatc and build min heap (x, f, size)
    curheap = creat min heap (size)   22BCE3939
    for i from 0 to size - 1
        cur heap . array [i] = newnode (x[i], f[i])
    cur heap . size = size
    build minheap ( curheap)
    return curheap
```

// creates a min heap of capacity equal to size and inserts all character of data[] in min heap.

```
function build huffman tree ( data, freq, size)
left, right, top
curheap = creat and build min heap
          ( data, freq, size)
while not isone (curheap)        // Iterate while size of heap is 1
    left = extract min (curheap)
    right = extract min (curheap)
    top = newnode ('$', left. freq + right. freq)
    top. left = left
    top. right = right
    insert_minheap (cur heap, top)
return extract min (curheap)
```

The main fn that builds huffman tree

```
function printCodes(root, arr, top)      // prints huffman codes from root of Huffman tree

    if root.left                         // Assign 0 to left edge
        arr[top] = 0                     and recur
        printCodes(root.left, arr, top + 1)

    if root.right                        // Assign 1 to right edge
        arr[top] = 1                     and recur
        printCodes(root.right, arr, top + 1)

    if isLeaf(root)                      // If this is a leaf node
        print root.data                  then it contains one
        printArr(arr, top)               of input characters,
                                         print the character
                                         and its code.

function huffmanCodes(data, freq, size)
    root = buildHuffmanTree(data, freq, size)
    arr[100], top = 0
    printCodes(root, arr, top)
# Driver Code
main()
read the input string from user
Obtain all the characters from the string
Obtain freq corresponding to the character
call huffmanCodes(data, freq, size)
```

22BCE3939

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
//leaf nodes
struct minheapnode{
  char data;
  int freq;
  struct minheapnode *left,*right;
};
//collection of leaf nodes
struct minheap{
  int size;
  int cap;
  struct minheapnode** array;
};
//creation of new leaf node to the min heap
struct minheapnode* newnode(char data,int freq){
  struct minheapnode* temp= (struct minheapnode *)malloc(sizeof(struct
minheapnode));
  temp->left=NULL;
  temp->right=NULL;
  temp->data=data;
  temp->freq=freq;
  return temp;
}
//creating a minheap of a certain capacity
struct minheap* createminheap(int cap){
  struct minheap* temp=(struct minheap*)malloc(sizeof(struct minheap));
  temp->size=0;
  temp->cap=cap;
  temp->array=(struct minheapnode*)malloc(cap*sizeof(struct minheapnode*));
}
//swaping two minheap nodes
void swapminheap(struct minheapnode** a,struct minheapnode** b){
  struct minheapnode *temp=*a;
  *a=*b;
  *b=temp;
}
//heapify function
void heapify(struct minheap* curheap,  int ind){
  int left=(2*ind)+1;
  int right=(2*ind)+2;
  int min=ind;
  if(left<curheap->size && curheap->array[left]->freq<curheap->array[min]->freq)
```

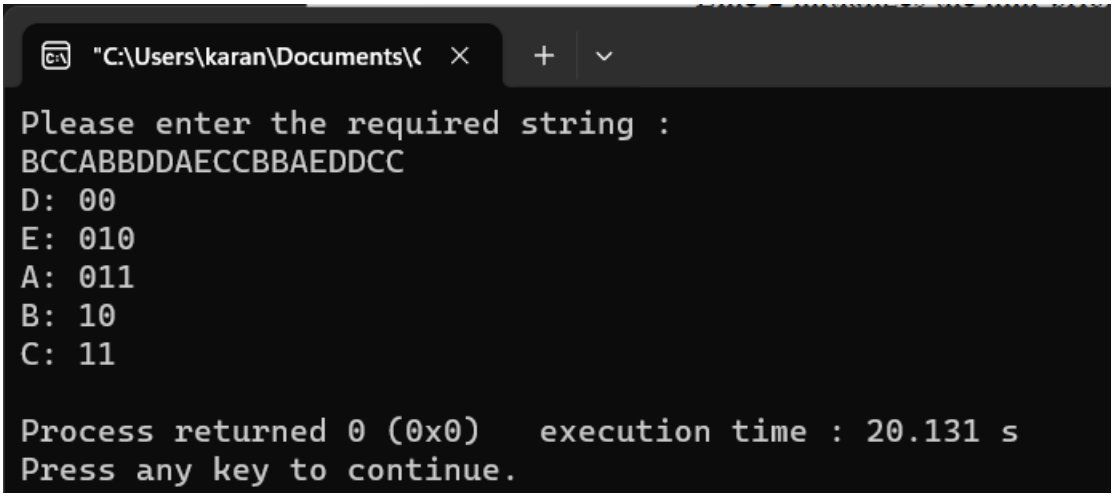
```
    min=left;
  if(right<curheap->size && curheap->array[right]->freq<curheap->array[min]-
>freq)
    min=right;
  if(min!=ind){
    swapminheap(&curheap->array[min],&curheap->array[ind]);
    heapify(curheap,min);
  }
}
//checks if the heap is of size 1
int isone(struct minheap* curheap){
  return(curheap->size==1);
}
//returns minimum value node from heap
struct minheapnode* extractmin(struct minheap* curheap){
  struct minheapnode* temp=curheap->array[0];
  curheap->array[0]=curheap->array[curheap->size-1];
  curheap->size--;
  heapify(curheap,0);
  return temp;
}
//insertion of newnode in minheap
void insertminheap(struct minheap* curheap,struct minheapnode* curnode){
  curheap->size++;
  int i=curheap->size-1;
  while(i && curnode->freq<curheap->array[(i-1)/2]->freq){
    curheap->array[i]=curheap->array[(i-1)/2];
    i=(i-1)/2;
  }
  curheap->array[i]= curnode;
}
//building minheap
void buildminheap(struct minheap* curheap){
  int n=curheap->size-1;
  int i;
  for(i=(n-1)/2;i>=0;--i){
    heapify(curheap,i);
  }
}
//function to print an array
void printarr(int arr[],int n){
  int i;
  for(i=0;i<n;i++){
    printf("%d",arr[i]);
  }
```

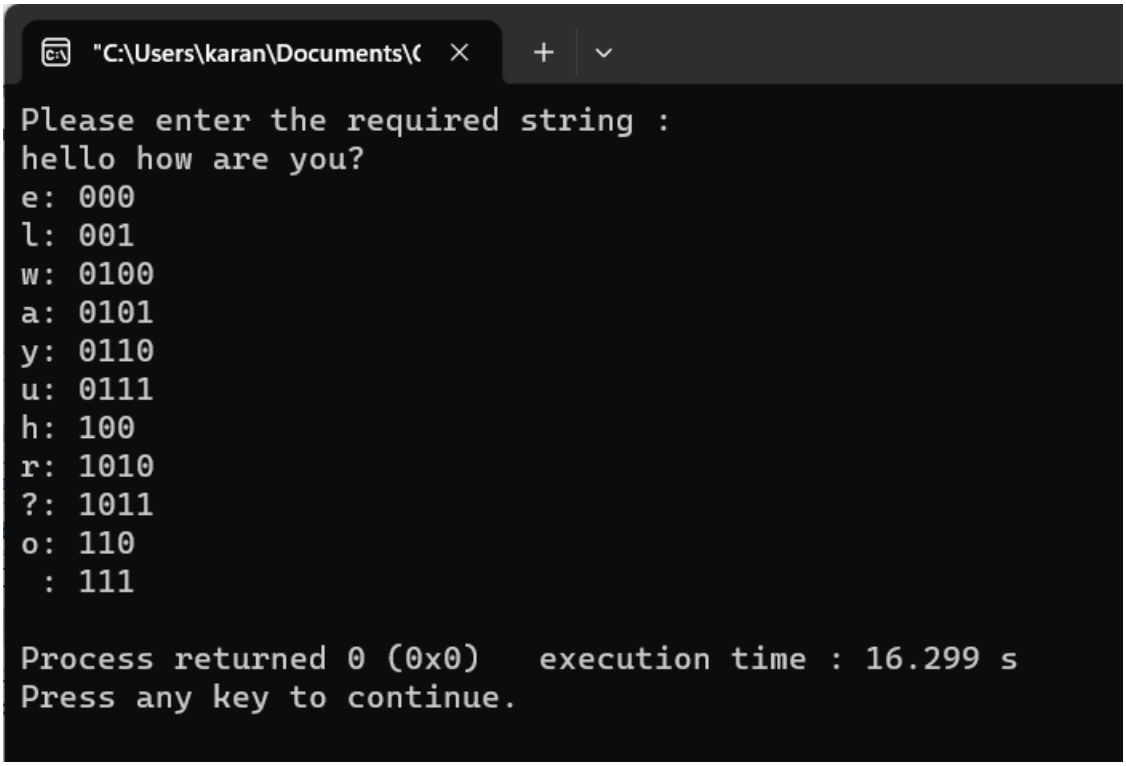
```
  printf("\n");
}
//check if leaf node
int isleaf(struct minheapnode* root){
  return !(root->left) && !(root->right);
}
//create and build minheap
struct minheap* createandbuildminheap(char data[],int freq[],int size){
  struct minheap* curheap=createminheap(size);
  for(int i=0;i<size;i++){
    curheap->array[i]=newnode(data[i],freq[i]);
  }
  curheap->size=size;
  buildminheap(curheap);
  return curheap;
}
//building huffmatree
struct minheapnode* buildhuffmantree(char data[],int freq[],int size){
  struct minheapnode *left,*right,*top;
  struct minheap* curheap=createandbuildminheap(data,freq,size);
  while(!isone(curheap)){
    left=extractmin(curheap);
    right=extractmin(curheap);
    top=newnode('$',left->freq+right->freq);
    top->left=left;
    top->right=right;
    insertminheap(curheap,top);
  }
  return extractmin(curheap);
}
//printing codes
void printcodes(struct minheapnode* root,int arr[],int top){
  if(root->left){
    arr[top]=0;
    printcodes(root->left,arr,top+1);
  }
  if(root->right){
    arr[top]=1;
    printcodes(root->right,arr,top+1);
  }
  if(isleaf(root)){
    printf("%c: ",root->data);
    printarr(arr,top);
  }
}
```

```c
//huffman codes
void huffmancodes(char data[],int freq[],int size){
  struct minheapnode* root= buildhuffmantree(data,freq,size);
  int arr[100],top=0;
  printcodes(root,arr,top);
}

int visited(char str[],char a,int j){
  int i=0;
  while(str[i]!='\0' && i<j){
    if(a==str[i])
      return 1;
    i++;
  }
  return 0;
}

int frequency(char str[],char a){
  int i=0;int freq=0;
  while(str[i]!='\0'){
    if(a==str[i])
      freq++;
    i++;
  }
  return freq;
}

void bubbleSort(char ch[], int freq[], int n) {
  for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
      if (freq[j] > freq[j + 1]) {
        // Swap frequencies
        int tempFreq = freq[j];
        freq[j] = freq[j + 1];
        freq[j + 1] = tempFreq;

        // Swap characters
        char tempChar = ch[j];
        ch[j] = ch[j + 1];
        ch[j + 1] = tempChar;
      }
    }
  }
}
int main(){
```

```
  char arr[100];
  char ch[100];
  int freq[100]={0};
  printf("Please enter the required string : \n");
  fgets(arr,sizeof(arr),stdin);
  int i=0;int j=0;
  while(arr[i]!='\0'){
    if(!visited(arr,arr[i],i)){
      ch[j]=arr[i];
      freq[j]=frequency(arr,arr[i]);
      j++;
    }
    i++;
  }
  char ch_new[j-1];
  int freq_new[j-1];
  for(i=0;i<j-1;i++){
    ch_new[i]=ch[i];
    freq_new[i]=freq[i];
  }
  bubbleSort(ch_new,freq_new,j-1);
  int size = sizeof(ch_new);
  huffmancodes(ch_new,freq_new,size);
  return 0;
}
```

Output:


"C:\Users\karan\Documents\(
Please enter the required string :
BCCABBDDAECCBBAEDDCC
D: 00
E: 010
A: 011
B: 10
C: 11
Process returned 0 (0x0)   execution time : 20.131 s
Press any key to continue.


"C:\Users\karan\Documents\(
Please enter the required string :
hello how are you?
e: 000
l: 001
w: 0100
a: 0101
y: 0110
u: 0111
h: 100
r: 1010
?: 1011
o: 110
 : 111
Process returned 0 (0x0)   execution time : 16.299 s
Press any key to continue.

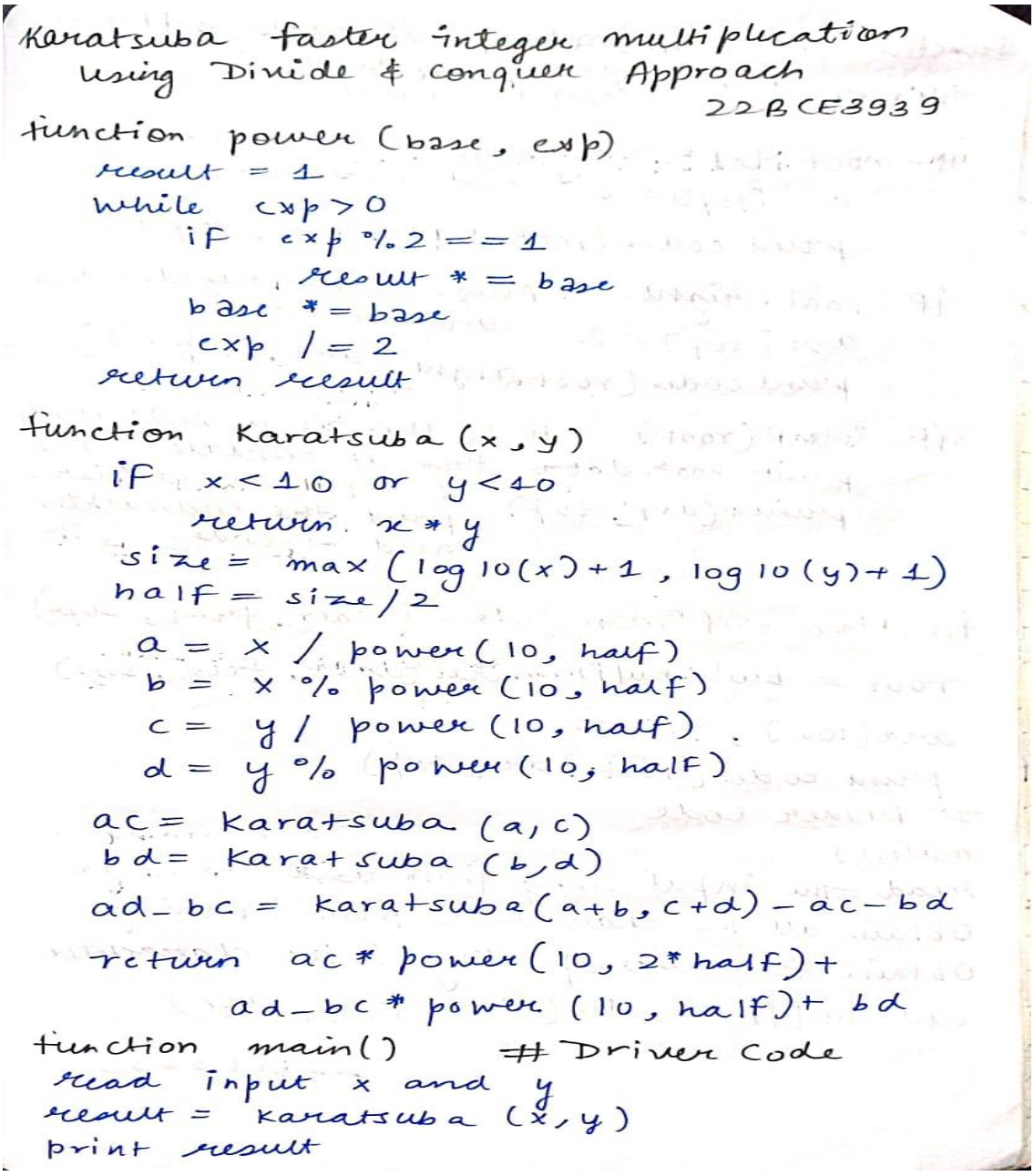
Question 3:

Karatsuba fast integer multiplication problem using Divide and conquer approach.

Problem Statement:

Consider multiplying two numbers, with their maximum length being n.

Pseudocode:



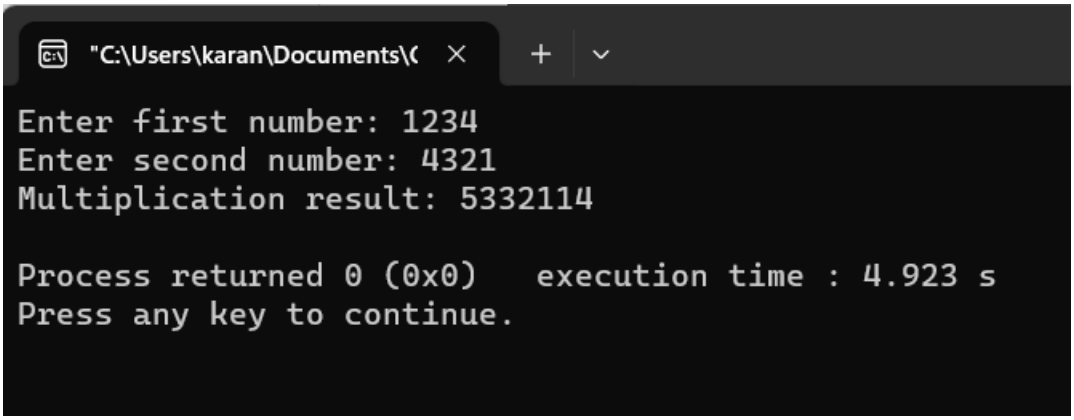
```
Karatsuba faster integer multiplication
   using Divide & conquer Approach
                                    22BCE3939
function power (base, exp)
    result = 1
    while exp > 0
        if exp % 2 == 1
            result *= base
        base *= base
        exp /= 2
    return result

function Karatsuba (x, y)
    if x < 10 or y < 10
        return x * y
    size = max (log10(x) + 1, log10(y) + 1)
    half = size / 2

     a = x / power(10, half)
     b = x % power(10, half)
     c = y / power(10, half)
     d = y % power(10, half)

    ac = karatsuba (a, c)
    bd = karatsuba (b, d)
    ad_bc = karatsuba(a+b, c+d) - ac - bd

    return ac * power(10, 2*half) +
           ad_bc * power (10, half) + bd
function main()      # Driver code
 read input x and y
 result = karatsuba (x, y)
 print result
```

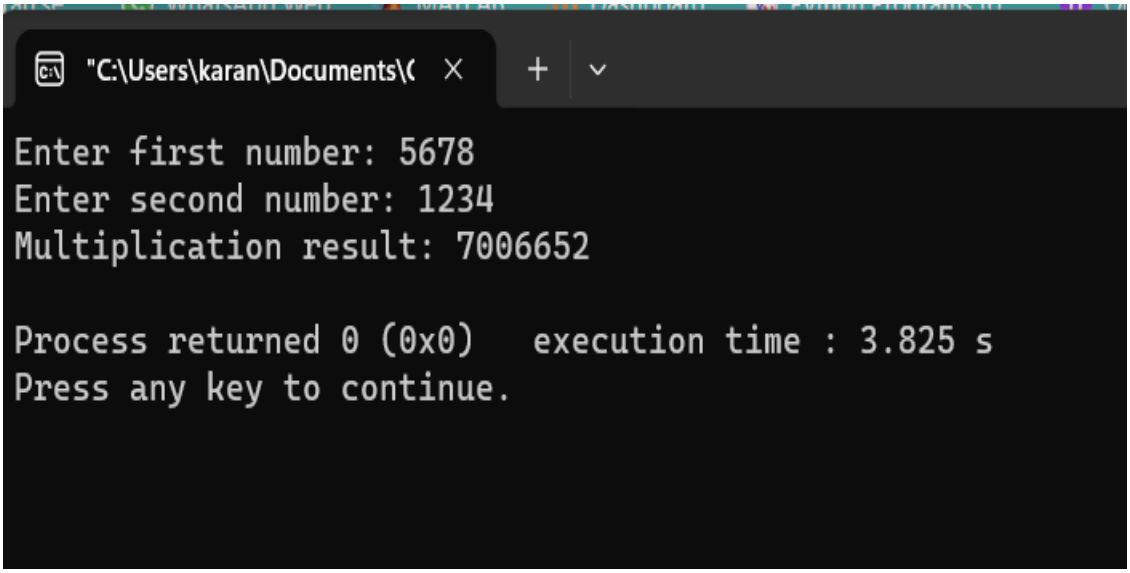
Source Code:

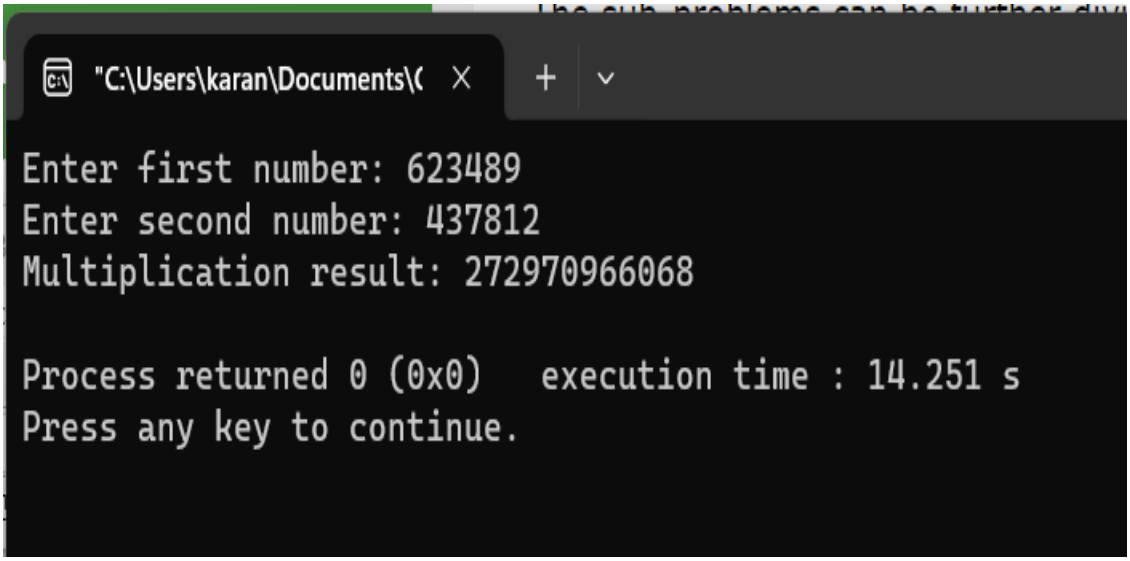
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

long long int power(int base, int exp) {
    long long int result = 1;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result *= base;
        }
        base *= base;
        exp /= 2;
    }
    return result;
}

long long int karatsuba(long long int x, long long int y) {
    if (x < 10 || y < 10) {
        return x * y;
    }

    int size = fmax(log10(x) + 1, log10(y) + 1);
    int half = size / 2;

    long long int a = x / power(10, half);
    long long int b = x % power(10, half);
    long long int c = y / power(10, half);
    long long int d = y % power(10, half);

    long long int ac = karatsuba(a, c);
    long long int bd = karatsuba(b, d);
    long long int ad_bc = karatsuba(a + b, c + d) - ac - bd;

    return ac * power(10, 2 * half) + ad_bc * power(10, half) + bd;
}

int main() {
    long long int x, y;

    printf("Enter first number: ");
    scanf("%lld", &x);

    printf("Enter second number: ");
```

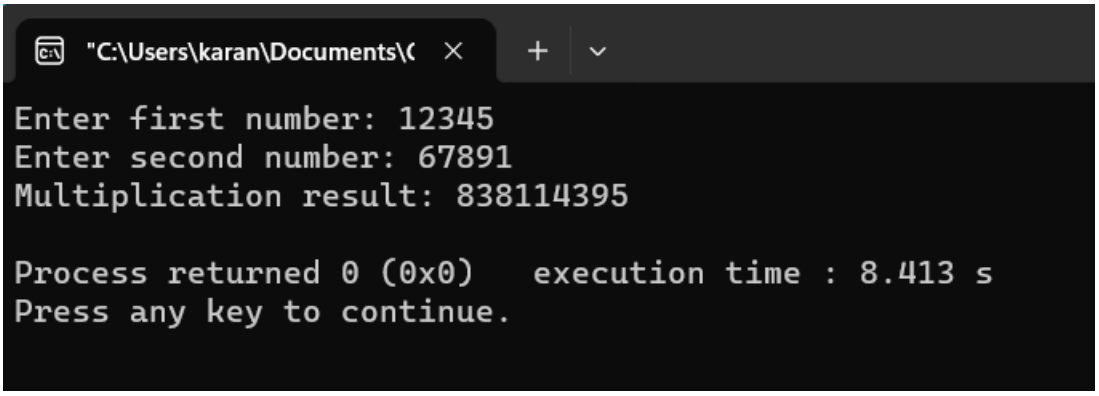
```
    scanf("%lld", &y);

    long long int result = karatsuba(x, y);

    printf("Multiplication result: %lld\n", result);

    return 0;
}
```

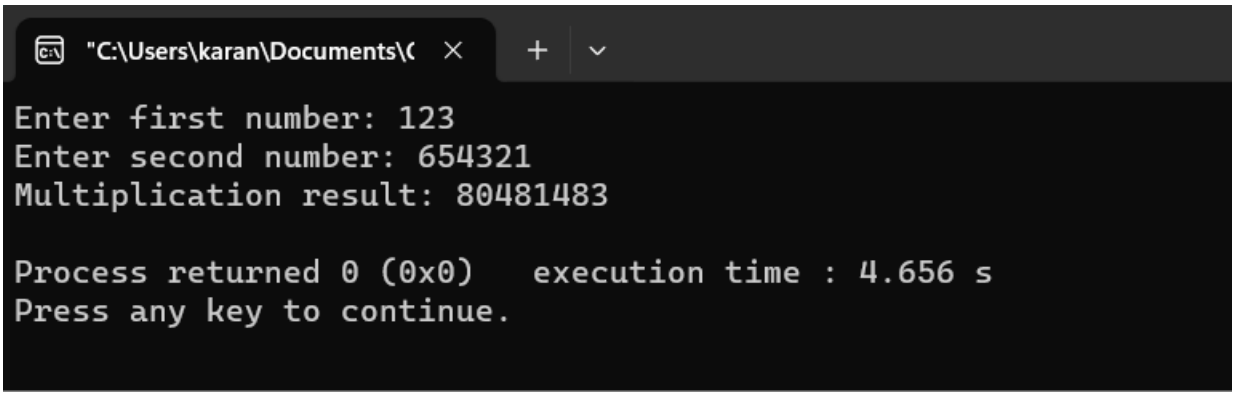
Output:


"C:\Users\karan\Documents\(
Enter first number: 1234
Enter second number: 4321
Multiplication result: 5332114
Process returned 0 (0x0) execution time : 4.923 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter first number: 5678
Enter second number: 1234
Multiplication result: 7006652
Process returned 0 (0x0) execution time : 3.825 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter first number: 623489
Enter second number: 437812
Multiplication result: 272970966068
Process returned 0 (0x0) execution time : 14.251 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter first number: 12345
Enter second number: 67891
Multiplication result: 838114395
Process returned 0 (0x0)   execution time : 8.413 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter first number: 123
Enter second number: 654321
Multiplication result: 80481483
Process returned 0 (0x0)   execution time : 4.656 s
Press any key to continue.

Question 4:

Maximum Subarray problem using Divide and conquer approach.

Problem Statement:

Given an integer array, find the maximum sum among all subarrays possible. Also return the starting and ending position of the subarray.

Array may contain negative and positive numbers which makes this a difficult
problem.

If all the array entries were positive, then the maximum-subarray problem would present no challenge, since the entire array would give the greatest sum.

Example: input array = [-3, 2, 3, 4, -4, -1]. Here maximum subarray is [2,3,4].
Hence maximum subarray sum is 9.

Pseudocode:

```
Maximum Subarray problem using
Divide & Conquer Approach
                                    22BCE3939

function MaxCrossingSum(arr, low, mid,
                        high)
left_sum = NEGATIVE_INFINITY
Set sum to 0
Initialize leftIndex to mid
for i from mid down to low
    sum += arr[i] // Add current elem
                     to sum
    if sum > leftsum
        leftsum = sum // update leftsum
        leftIndex = i & leftIndex if
                    sum is greater
rightsum = NEGATIVE_INFINITY
sum = 0
rightIndex = mid+1 // Initialize rightIndex
                      to mid+1
for i from mid+1 to high
    sum += arr[i]
    if sum > rightsum // update rightsum
        rightsum = sum & rightindex
        right index = i if sum is greater

return { start : leftIndex
         end : right index
         sum : left sum + rightsum }
// return the result
```

22BCE3939

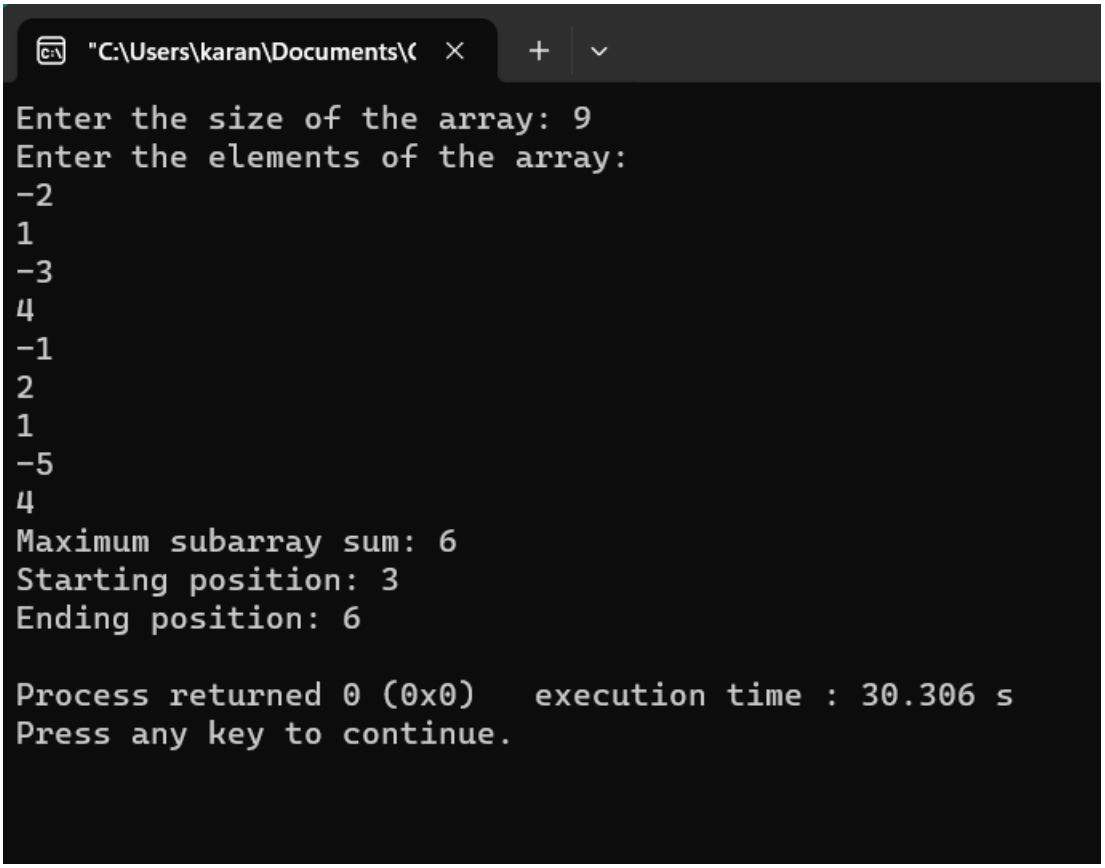
```
function MaxSubbarraySum(arr, low, high)
    if low == high
        return arr[low] // Base case: single element
    mid = floor((low + high)/2)
    left = MaxSubbarraySum(arr, low, mid)
        └→ // recursively find max on left
    right = MaxSubbarraySum(arr, mid+1, high)
        └→ // recursively find max on right
    cross = MaxCrossingSum(arr, low, mid, high)
        └→ // Finding max cross sum
    return maximum(left, right, cross)
        // compare & return max of left, right & cross.
# Driver Code
read the no. of elements (n)
initialize the array arr with size n
read the elements using for loop
result = MaxSubbarraySum(arr, 0, n-1)
print "Maximum subarray sum": result.sum
print "start pos": result.start
print "end pos": result.end
```

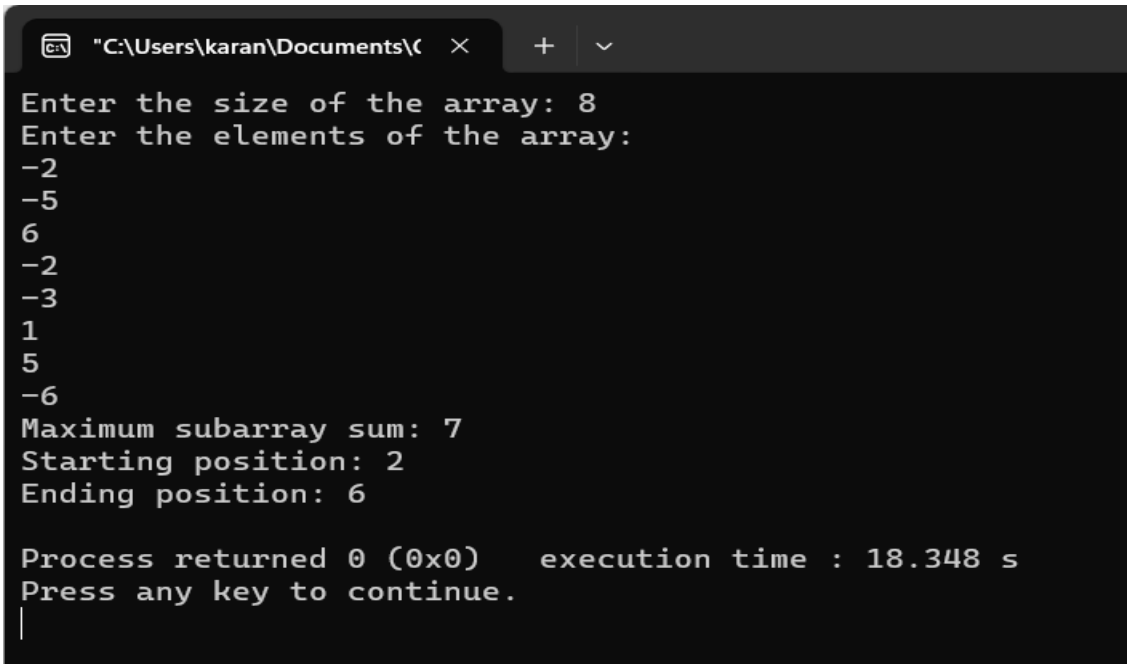
Source Code:

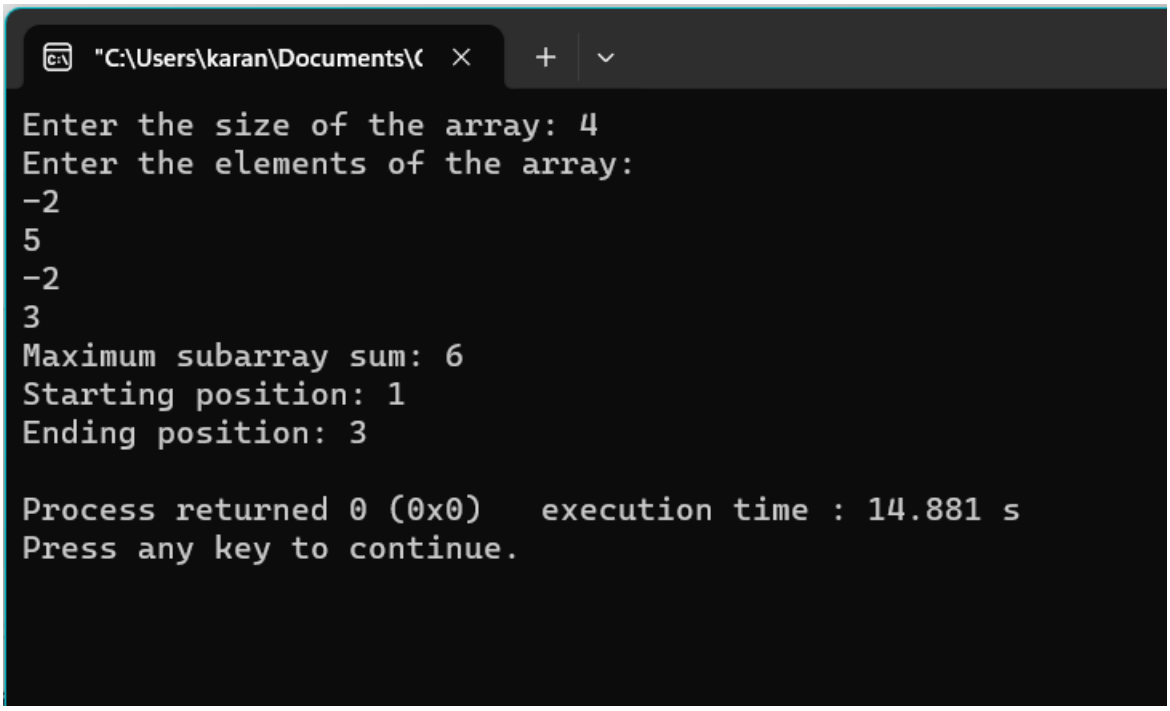
```c
#include <stdio.h>
#include <limits.h>

typedef struct {
    int start;
    int end;
    int sum;
} MaxSubarray;

MaxSubarray maxCrossingSum(int arr[], int low, int mid, int high) {
    MaxSubarray result;

    int leftSum = INT_MIN;
    int sum = 0;
    int leftIndex = mid;

    for (int i = mid; i >= low; i--) {
        sum += arr[i];
        if (sum > leftSum) {
            leftSum = sum;
            leftIndex = i;
        }
    }

    int rightSum = INT_MIN;
    sum = 0;
    int rightIndex = mid + 1;

    for (int i = mid + 1; i <= high; i++) {
        sum += arr[i];
        if (sum > rightSum) {
            rightSum = sum;
            rightIndex = i;
        }
    }

    result.start = leftIndex;
    result.end = rightIndex;
    result.sum = leftSum + rightSum;

    return result;
}
```

```c
MaxSubarray maxSubarraySum(int arr[], int low, int high) {
    MaxSubarray result;

    if (low == high) {
        result.start = low;
        result.end = high;
        result.sum = arr[low];
        return result;
    }

    int mid = (low + high) / 2;

    MaxSubarray left = maxSubarraySum(arr, low, mid);
    MaxSubarray right = maxSubarraySum(arr, mid + 1, high);
    MaxSubarray cross = maxCrossingSum(arr, low, mid, high);

    if (left.sum >= right.sum && left.sum >= cross.sum) {
        return left;
    } else if (right.sum >= left.sum && right.sum >= cross.sum) {
        return right;
    } else {
        return cross;
    }
}

int main() {
    int n;

    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];

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

    MaxSubarray maxSub = maxSubarraySum(arr, 0, n - 1);

    printf("Maximum subarray sum: %d\n", maxSub.sum);
    printf("Starting position: %d\n", maxSub.start);
    printf("Ending position: %d\n", maxSub.end);

    return 0;}
```

Output:


"C:\Users\karan\Documents\(
Enter the size of the array: 9
Enter the elements of the array:
-2
1
-3
4
-1
2
1
-5
4
Maximum subarray sum: 6
Starting position: 3
Ending position: 6
Process returned 0 (0x0)   execution time : 30.306 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter the size of the array: 8
Enter the elements of the array:
-2
-5
6
-2
-3
1
5
-6
Maximum subarray sum: 7
Starting position: 2
Ending position: 6
Process returned 0 (0x0)   execution time : 18.348 s
Press any key to continue.


"C:\Users\karan\Documents\(
Enter the size of the array: 4
Enter the elements of the array:
-2
5
-2
3
Maximum subarray sum: 6
Starting position: 1
Ending position: 3
Process returned 0 (0x0)   execution time : 14.881 s
Press any key to continue.

Question 5:

N-Queens problem using Backtracking approach.

Problem Statement:

N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.

It can be seen that for n =1, the problem has a trivial solution, and no solution exists for n =2 and n =3. So first we will consider the 4 queens problem and then generate it to n - queens problem.

Pseudo-code:

N-Queens Problem using Backtracking Technique

22BCE3939

```
// A utility function to check if a queen can be placed on board[row][col]. Note that this fn is called when queens are placed in "col". so we need to check only left side for attacking queens

function isSafe(row, col, n)
    for (i) from 0 to col-1        // Checking row on left side
        if board[row][i] is 1
            return false
    for i from row, j from col to 0,
        decrement i and j          // checking upper diagonal on left side
        if board[i][j] = 1
            return false
    for i from row, j from col to 0,
        increment i and decrement j until i<n
        if board[i][j] = 1         // checking lower diagonal on left side.
            return false
    return true
function printSol^n (n)
    for i from 0 to n-1
        for j from 0 to n-1
            if board[i][j] = 1
                print (i,j)
            else
                print "(.)"
        print new line
```

// A recursive Utility Function to solve N queens problem. 22BCE3939

function SolveNQueens (col, n)

if col >= n // Base case: All queens are placed.

return true

for (i) from 0 to n-1

if isSafe (i, col, n) // Check if queen can be placed

Set board [i][col] to 1 // Place this queen

if SolveNQueens (col+1, n) // Recur to place rest of queens

return true

board [i][col] = 0 // If placing queen in board [i][col] is not leading to a soln, BACKTRACK

return false

// Driver Code

Function main()

read n from user

if n < 4 or n > N

print "Invalid Board size"

Initialise board of size n x n from user with zeroes.

if call SolveNQueens (0,n) is false

printf "Soln doesn't Exist"

return 1

call printSoln (n)

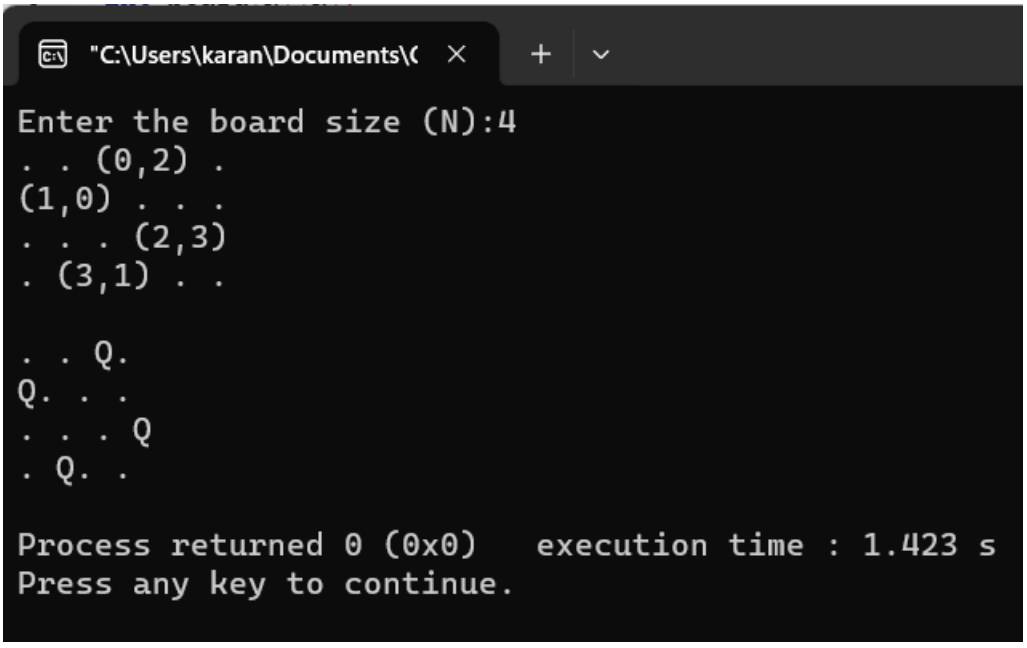
Source code:

```
#define N 15
#include <stdbool.h>
#include <stdio.h>
int board[N][N];
void printSolution(int n)
{
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if(board[i][j])
                printf("(%d,%d) ",i,j);
            else
                printf(". ");
        }
        printf("\n");
    }
}
void printb(int n)
{
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if(board[i][j])
                printf("Q");
            else
                printf(". ");
        }
        printf("\n");
    }

}
bool isSafe(int row, int col, int n)
{
    int i, j;
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;

    return true;
```

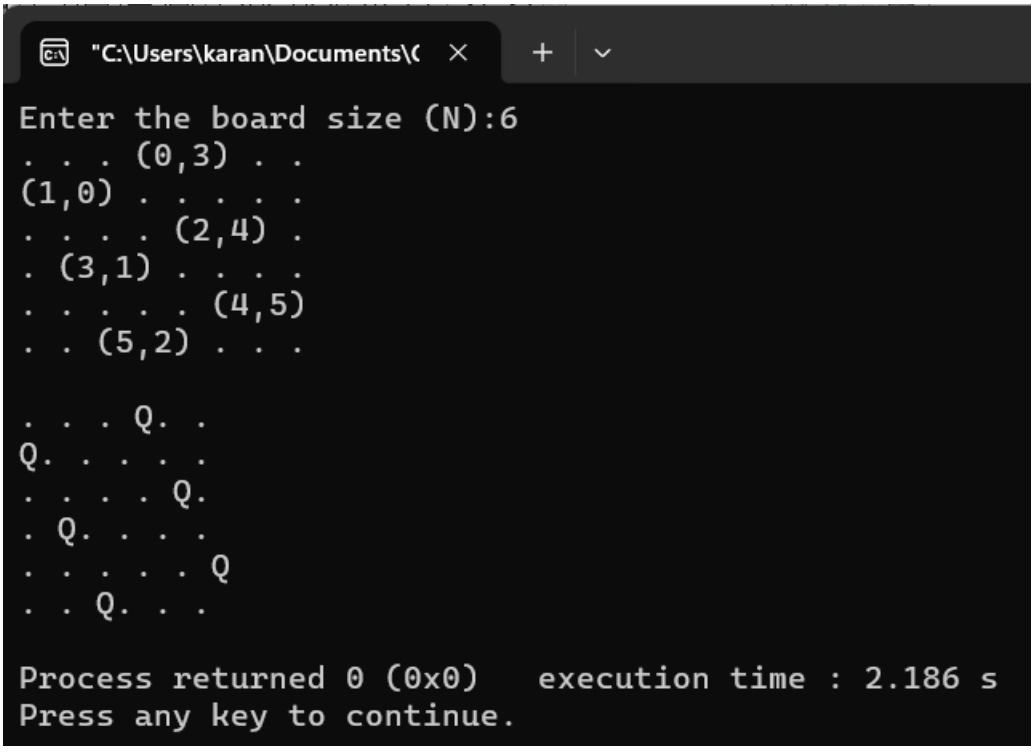
```
}
bool solveNQueens(int col,int n)
{
    if (col >= n)
        return true;

    for (int i = 0; i < n; i++) {

        if (isSafe(i,col,n)) {

            board[i][col] = 1;

            if (solveNQueens(col + 1,n))
                return true;

            board[i][col] = 0;
        }
    }
    return false;
}
int main()
{
    int n;
    printf("Enter the board size (N):");
    scanf("%d",&n);
    if(n < 4 || n > N)
    {
        printf("Invalid Board size. Please Enter a value greater than 4");
        return 1;
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            board[i][j]=0;
        }
    }
    if (solveNQueens(0,n) == false) {
        printf("Solution does not exist");
        return false;
    }
     printSolution(n);
     printf("\n");
     printb(n);
    return 0;}
```

Output:

N=4


"C:\Users\karan\Documents\(
Enter the board size (N):4
. . (0,2) .
(1,0) . . .
. . . (2,3)
. (3,1) . .
. . Q.
Q. . .
. . . Q
. Q. .
Process returned 0 (0x0) execution time : 1.423 s
Press any key to continue.

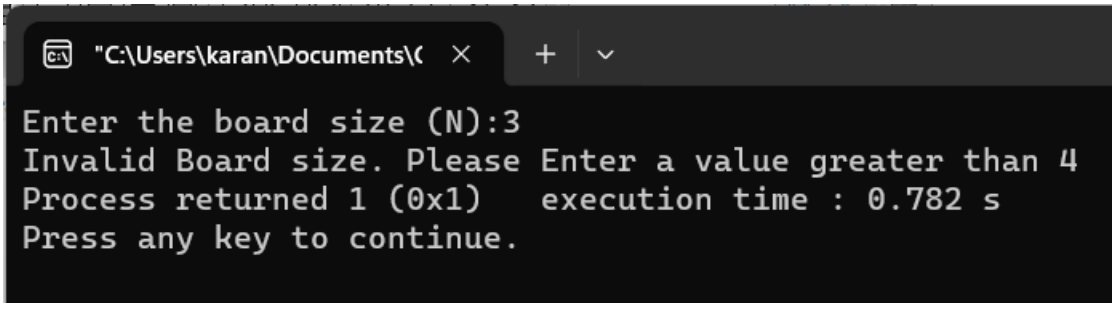
N=6


"C:\Users\karan\Documents\(
Enter the board size (N):6
. . . (0,3) . .
(1,0) . . . . .
. . . . (2,4) .
. (3,1) . . . .
. . . . . (4,5)
. . (5,2) . . .
. . . Q. .
Q. . . . .
. . . . Q.
. Q. . . .
. . . . . Q
. . Q. . .
Process returned 0 (0x0) execution time : 2.186 s
Press any key to continue.

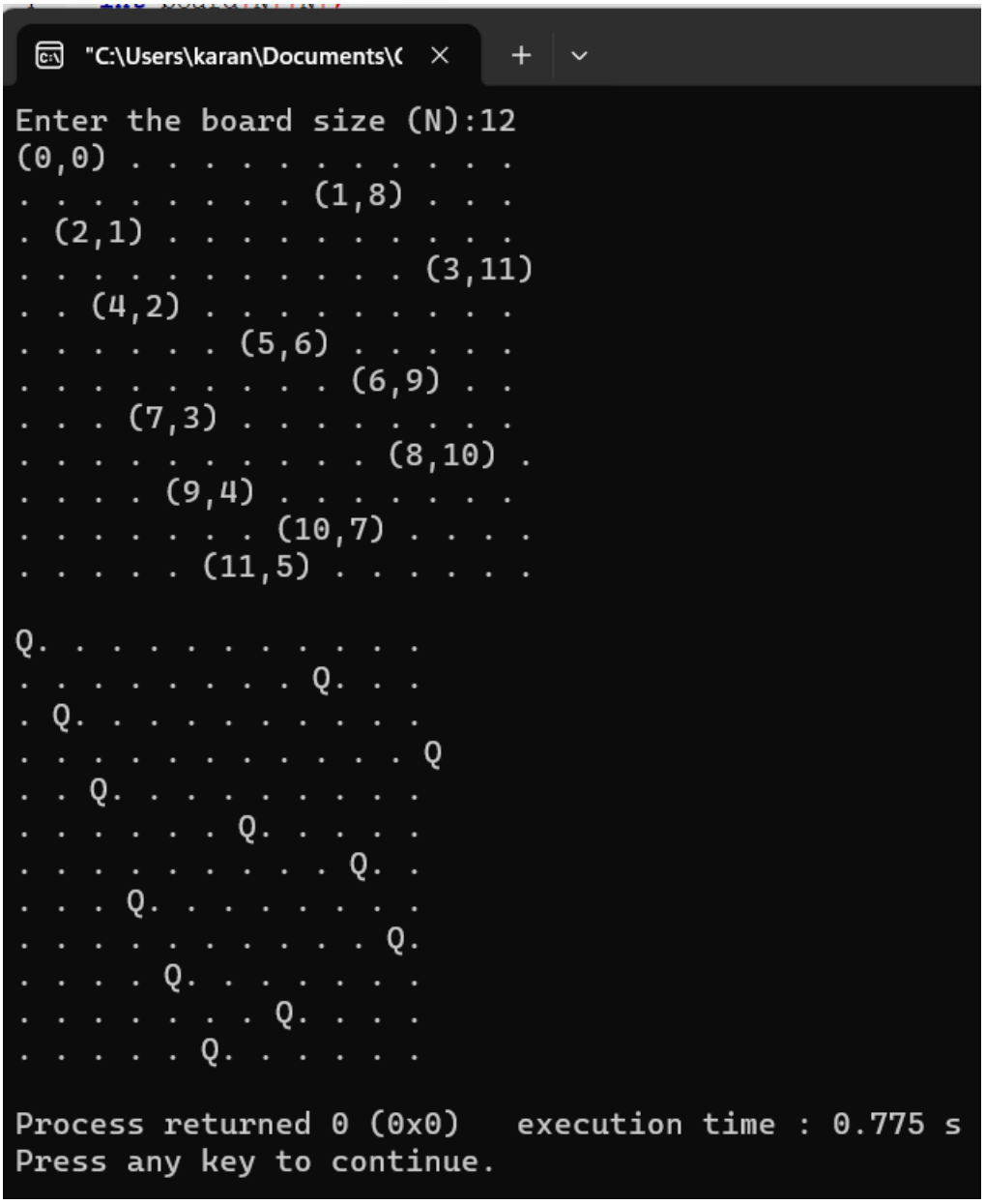
N=9


"C:\Users\karan\Documents\(
Enter the board size (N):9
(0,0) . . . . . .
. . . . (1,4) . . . .
. (2,1) . . . . . .
. . . . . (3,5) . . .
. . . . . (4,8)
. . (5,2) . . . . . .
. . . . . (6,7) .
. . . (7,3) . . . . .
. . . . . (8,6) . .
Q. . . . . .
. . . . Q. . . .
. Q. . . . . .
. . . . . Q. . .
. . . . . Q
. . Q. . . . . .
. . . . . Q.
. . . Q. . . . .
. . . . . Q. .
Process returned 0 (0x0) execution time : 0.876 s
Press any key to continue.

N=3


"C:\Users\karan\Documents\(
Enter the board size (N):3
Invalid Board size. Please Enter a value greater than 4
Process returned 1 (0x1) execution time : 0.782 s
Press any key to continue.

N=12


"C:\Users\karan\Documents\(
Enter the board size (N):12
(0,0) . . . . . .
. . . . . (1,8) . . .
. (2,1) . . . . . .
. . . . . (3,11)
. . (4,2) . . . . . .
. . . . . (5,6) . . . . .
. . . . . (6,9) . .
. . . (7,3) . . . . . .
. . . . . (8,10) .
. . . . (9,4) . . . . . .
. . . . . (10,7) . . . .
. . . . . (11,5) . . . . . .
Q. . . . . .
. . . . . Q. . .
. Q. . . . . .
. . . . . Q
. . Q. . . . . .
. . . . . Q. . . . .
. . . . . Q. .
. . . Q. . . . . .
. . . . . Q.
. . . . Q. . . . . .
. . . . . Q. . . .
. . . . . Q. . . . . .
Process returned 0 (0x0) execution time : 0.775 s
Press any key to continue.