Lab Report: 03 (Evaluation)

Sub Code: CSE246

Sec: 03

Course Title: Algorithm

Semester: Fall-2021

Course Instructor: Jeson Ahammed Ovi, Senior Lecturer, Dept. of Computer Science & Engineering.

Submitted To:

Jeson Ahammed Ovi

Senior Lecturer, Dept. of

Computer Science & Engineering

Submitted By:

Name	: Kazi Sifat Al Maksud
ID	: 2019-3-60-050

Answer to the Question on: 01

Problem Statement: 1. Activity selection problem: Given a set of activities, along with the starting and finishing time of each activity, find the maximum number of activities performed by a single person assuming that a person can only work on a single activity at a time.

Example:

3,7 2,4 5,8 6,9 1,11 10,12 0,3

Output: 3

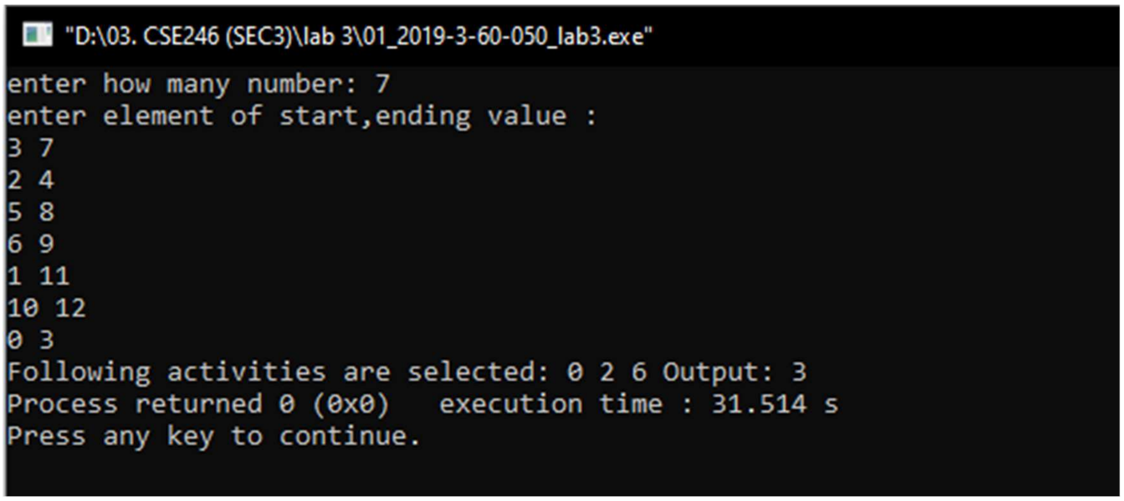
Code:

```
#include<stdio.h>

struct ass{
int start;
int end;
}arr[100],tmp;
int main()
{
    int i,j;
        int n,x,y;
        printf("enter how many number: ");
        scanf("%d",&n);
        printf("enter element of start,ending value : \n");
        for(int i =0;i<n;i++){
         scanf("%d",&x);
          scanf("%d",&y);
         arr[i].start =x;
         arr[i].end =y;
        }


    for(int i= 0;i<n-1;i++){
        for(int j=0;j<n-1-i;j++){
            if (arr[j].end >arr[j+1].end){
                tmp = arr[j];
                arr[j] =arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
    int c=0;
        printf ("Following activities are selected: ");
        i = 0;
        printf("%d ", i);
        for (j = 1; j < n; j++)
        {
        if (arr[j].start >= arr[i].end)
        {
                printf ("%d ", j);
                i = j;
                c++;
        }
        }
        printf("Output: %d",c+1);
}
```

Output:



```
"D:\03. CSE246 (SEC3)\lab 3\01_2019-3-60-050_lab3.exe"
enter how many number: 7
enter element of start,ending value :
3 7
2 4
5 8
6 9
1 11
10 12
0 3
Following activities are selected: 0 2 6 Output: 3
Process returned 0 (0x0)   execution time : 31.514 s
Press any key to continue.
```

Theory:

The greedy choice is to always pick the next activity whose finish time is least among the remaining activities and the start time is more than or equal to the finish time of the previously selected activity. We can sort the activities according to their finishing time so that we always consider the next activity as minimum finishing time activity.

1) Sort the activities according to their finishing time

2) Select the first activity from the sorted array and print it.

3) Do the following for the remaining activities in the sorted array.

a) If the start time of this activity is greater than or equal to the finish time of the previously selected activity then select this activity and print it.

Discretion:

1. At first, I try to solve is problem by array. I use two arrays. After that when I see two arrays need to swap for solve this problem. That time I decided to use structure for solving is problem.
2. Overclocking starting value and ending value. For this reason, I consume too much time for to solve this problem. I use for loop for to solve is problem. I declare a counter for staring value. Then, whenever I see staring value is greater then ending value that time I print the starting value after that I set the j value in the ''I" variable.

<u>Problem Statement: 2.</u> Minimize the sum of Product: You are given two arrays, A and B, of equal size N. The task is to

find the minimum value of A[0] * B[0] + A[1] * B[1] +...+ A[N-1] * B[N-1], where shuffling of

elements of arrays A and B is allowed.

Example 1:

Input: N = 3 A[] = {3, 1, 1} B[] = {6, 5, 4}

Output: 23

Explanation: 1*6+1*5+3*4 = 6+5+12 = 23 is the minimum sum

Example 2:

Input: N = 5 A[] = {6, 1, 9, 5, 4} B[] = {3, 4, 8, 2, 4}

Output: 80

Explanation: 2*9+3*6+4*5+4*4+8*1 =18+18+20+16+8 = 80 is the minimum sum

<u>Code:</u>

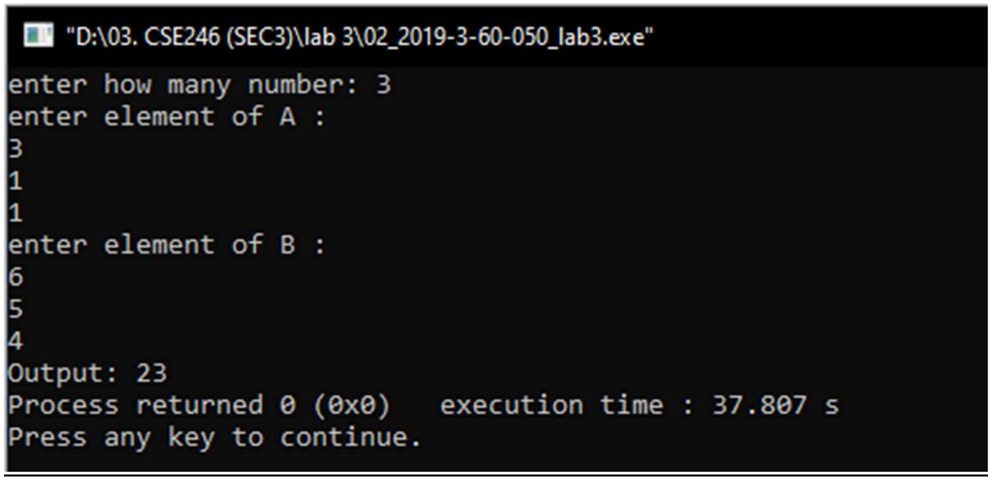
```
#include<stdio.h>

int main()
{
    int tmp;
        int n,x,y;
        int a[100],b[100];
        printf("enter how many number: ");
        scanf("%d",&n);
        printf("enter element of A : \n");
        for(int i =0;i<n;i++){
         scanf("%d",&x);
         a[i] =x;
        }
        printf("enter element of B : \n");
                for(int i =0;i<n;i++){
```

```
          scanf("%d",&y);
          b[i]= y;
         }
     for(int i= 0;i<n-1;i++){
         for(int j=0;j<n-1-i;j++){
             if (a[j] >a[j+1]){
                 tmp = a[j];
                 a[j] =a[j+1];
                 a[j+1] = tmp;
             }
         }
     }
     for(int i= 0;i<n-1;i++){
         for(int j=0;j<n-1-i;j++){
             if (b[j] <b[j+1]){
                 tmp = b[j];
                 b[j] =b[j+1];
                 b[j+1] = tmp;
             }
         }
     }
     int sum =0;
     for(int i=0;i<n;i++){
             int value = a[i]*b[i];
         sum = sum + value;
     }
     printf("Output: %d",sum);


}
```

Output:



"D:\03. CSE246 (SEC3)\lab 3\02_2019-3-60-050_lab3.exe"

```
enter how many number: 3
enter element of A :
3
1
1
enter element of B :
6
5
4
Output: 23
Process returned 0 (0x0)   execution time : 37.807 s
Press any key to continue.
```

Theory:

The idea is to multiply minimum element of one array to maximum element of another array. Algorithm to solve this problem:

1.Sort both the arrays A and B.

2.Traverse the array and for each element, multiply A[i] and B[n – i – 1] and add to the total.

Discretion:

1. In this problem I can't face any major problem.

   Is this problem I have two arrays. I swap first one in ascending order, other one swap in disbanding order. And then, I run a loop for multiply two arrays' elements. then, store the value in sum variable. After multiplication I print the sum variables.

Problem statement: 3. Assigning mice to hole: There are N Mice and N holes are placed in a straight line. Each hole can accommodate only 1 mouse. A mouse can stay at his position, move one step right from x to x + 1, or move one step left from x to x -1. Any of these moves consumes 1 minute. Assign mice to holes so that the time when the last mouse gets inside a hole is minimized.

Example:

Input: positions of mice are: 4 -4 2

Positions of holes are: 4 0 5

Output: 4

Assign mouse at position x = 4 to hole at

Position x = 4 : Time taken is 0 minutes

Assign mouse at position x=-4 to hole at

Position x = 0 : Time taken is 4 minutes

Assign mouse at position x=2 to hole at

Position x = 5 : Time taken is 3 minutes

After 4 minutes all of the mice are in the holes.

Since, there is no combination possible where

the last mouse's time is less than 4,

Answer = 4.

Input: positions of mice are: -10, -79, -79, 67, 93, -85, -28, -94

Positions of holes are: -2, 9, 69, 25, -31, 23, 50, 78

Output: 102

Code:

```
#include<stdio.h>

int main()

{

   int tmp;

            int n,x,y;

            int a[100],b[100];

            printf("enter how many number: ");

            scanf("%d",&n);

            printf("enter element of A : \n");

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

      scanf("%d",&x);

      a[i] =x;}

            printf("enter element of B : \n");

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

      scanf("%d",&y);

      b[i]= y;}

             for(int i= 0;i<n-1;i++){

      for(int j=0;j<n-1-i;j++){

         if (a[j] >a[j+1]){

            tmp = a[j];

            a[j] =a[j+1];

            a[j+1] = tmp;

         } } }

              for(int i= 0;i<n-1;i++){
```

```
        for(int j=0;j<n-1-i;j++){
          if (b[j] <b[j+1]){
            tmp = b[j];
            b[j] =b[j+1];
            b[j+1] = tmp;
          } }
           int max = 0;
           for(int i = 0; i < n; ++i)
           {
                   if (max <b[i] - a[i]){
                           max = a[i] - b[i];
                   }}
           printf("%d",max);
}
```

Output:

```
//error; // Couldn't solve
```

Theory: This problem can be solved using greedy strategy. We can put every mouse to its nearest hole to minimize the time. This can be done by sorting the positions of mice and holes. This allows us to put the ith mice to the corresponding hole in the holes list. We can then find the maximum difference between the mice and corresponding hole position.

In example 2, on sorting both the lists, we find that the mouse at position -79 is the last to travel to hole 23 taking time 102.sort mice positions (in any order) sort hole positions Loop i = 1 to N:

update ans according to the value of |mice(i) - hole(i)|. It should be maximum of all differences.

Proof of correctness: Let i1 < i2 be the positions of two mice and let j1 < j2 be the positions of two holes.

It suffices to show via case analysis that max (|i1-j1|, |i2-j2|) <= max(|i1-j2|, |i2-j1|),

where '|a - b|' represent absolute value of (a - b)

Since it follows by induction that every assignment can be transformed by a series of swaps into the sorted assignment, where none of these swaps increases the span.

== The End ==