Design and Analysis of Algorithms Lab Assessment-1

Name: Armaan Indani

Reg. No.: 22BCE3347

Slot : L41+42

Faculty: Dr. Joshva Devdas T – 16700

ASSESSMENT LAB 1

Started on	Friday, 26 January 2024, 6:20 AM
State	Finished
Completed on	Sunday, 4 February 2024, 10:51 PM
Time taken	9 days 16 hours

Question **1**
Correct
Marked out of 2.00
Flag question

Write C functions to analyze the 0/1 Knapsack Problem thus by implementing using Dynamic Programming(set method)

Given that n=5,

(P1,P2,P3,P4,P5)=(W1,W2,W3,W4,W5)=(7,4,3,5,2,9) and m=15

Note : profit is same as the weight of each item

For example:

Test	Input	Result

1	6
	15
	7 4 3 5 2 9
	7 4 3 5 2 9
		010011
		101100

Answer: (penalty regime: 0 %)

```
#include<stdio.h>

void knapsack(int n, char *s, int item, int p[n], int w[n],int m, int best)
{
    if(item>n)
    {
        return;
    }
    int wt = 0, pf = 0;
    for(int i=0; i<n; i++)
    {
        if(s[i] == '1')
        {
            wt+=w[i];
            pf+=p[i];
        }
    }

    if(wt>m)
    {
        return;
    }


    if(s[n-1] != '2')
    {
        if(pf>=best)
        {
            best = pf;
        }
        if(wt==m)
        {
            puts(s);
        }
        return;
```

```c
    }

    s[item] = '0';
    for(int i = item+1; i<n;i++)
    {s[i] = '2';}
    knapsack(n,s,item+1,p,w,m,best);


    s[item] = '1';
    for(int i = item+1; i<n;i++)
    {s[i] = '2';}
    knapsack(n,s,item+1,p,w,m,best);
}


int main()
{
    int n,m;
    scanf("%d",&n);
    scanf("%d", &m);

    int p[n],w[n];
    char s[n];

    for(int i=0; i<n; i++)
    {
        scanf("%d", &p[i]);
        s[i] = '2';
    }
    for(int i=0; i<n; i++)
    {
        scanf("%d", &w[i]);
    }

    knapsack(n,&s[0],0,p,w,m,0);
}
```

	Test	Input	Expected	Got	
✓	1	6 15 7 4 3 5 2 9 7 4 3 5 2 9	010011 101100	010011 101100	✓

Passed all tests! ✓

Question 2

Correct

Marked out of 2.00

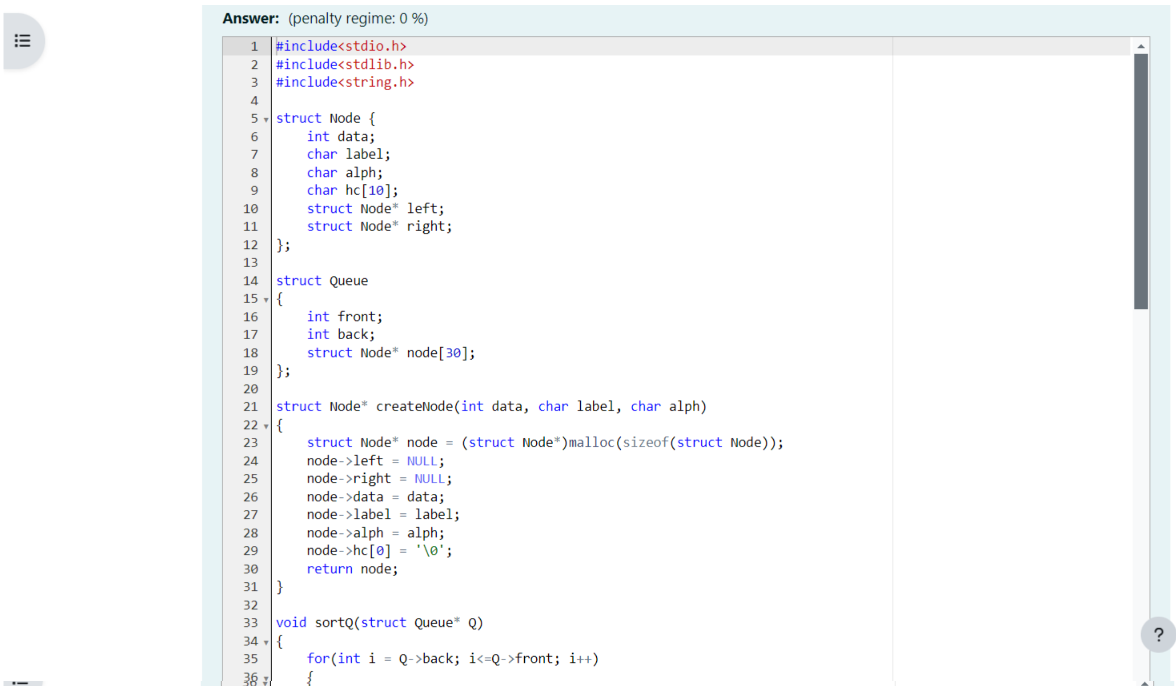
Flag question

Write a program to generate Huffman coding for the following Text given

ADAM IS IN MADAMS KITCHEN

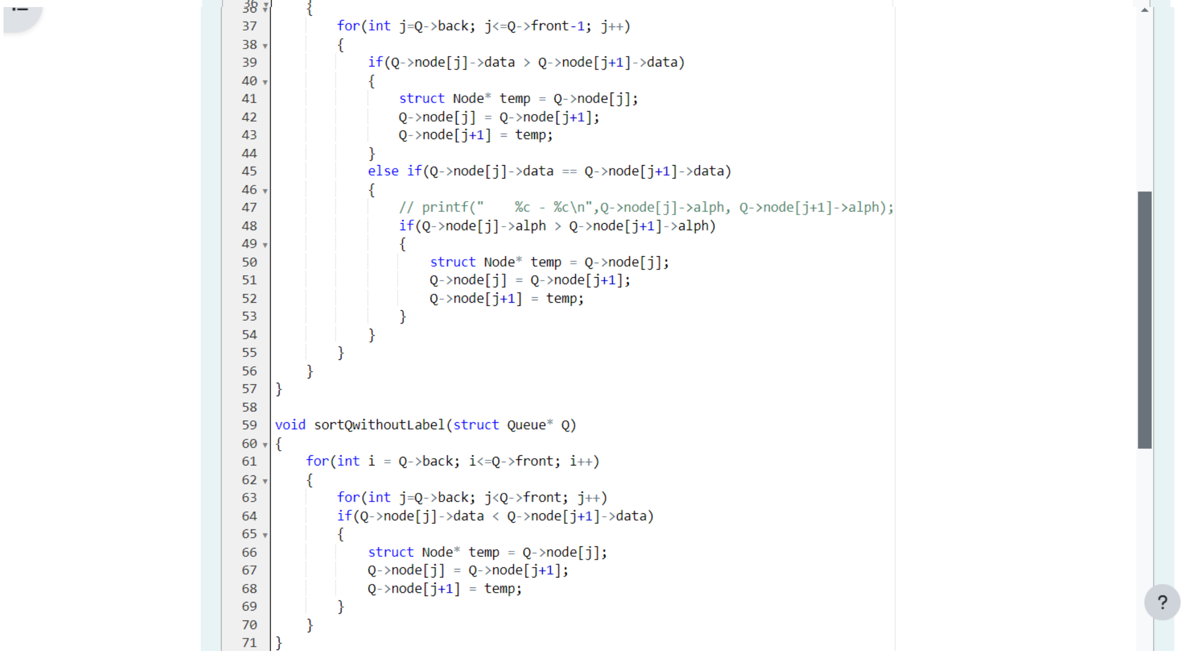
For example:

Test	Input	Result
1	ADAM IS IN MADAMS KITCHEN	FREQUENCY OF ALPHABETS A-4 M-3 I-3 D-2 S-2 N-2 K-1 T-1 C-1 H-1 E-1 HUFFMAN CODE IS C-11010 E-11011 H-0010 K-0011 T-1100 D-000 N-010 S-011 I-100 M-101 A-111

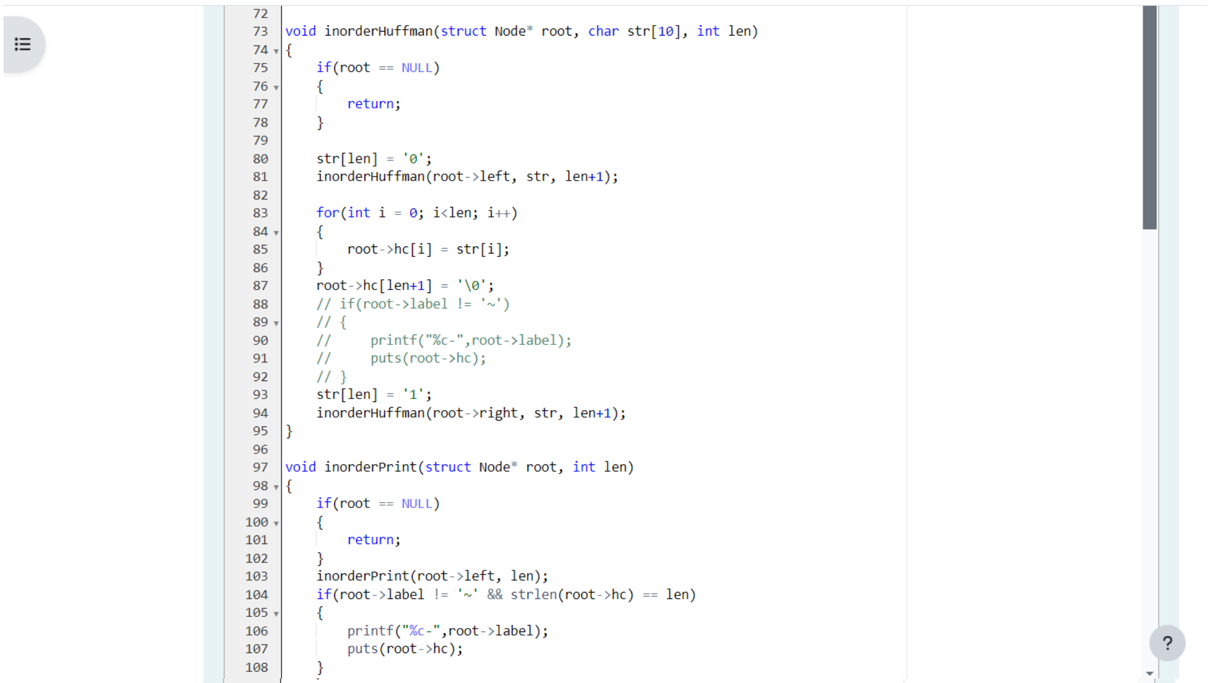


Answer: (penalty regime: 0 %)

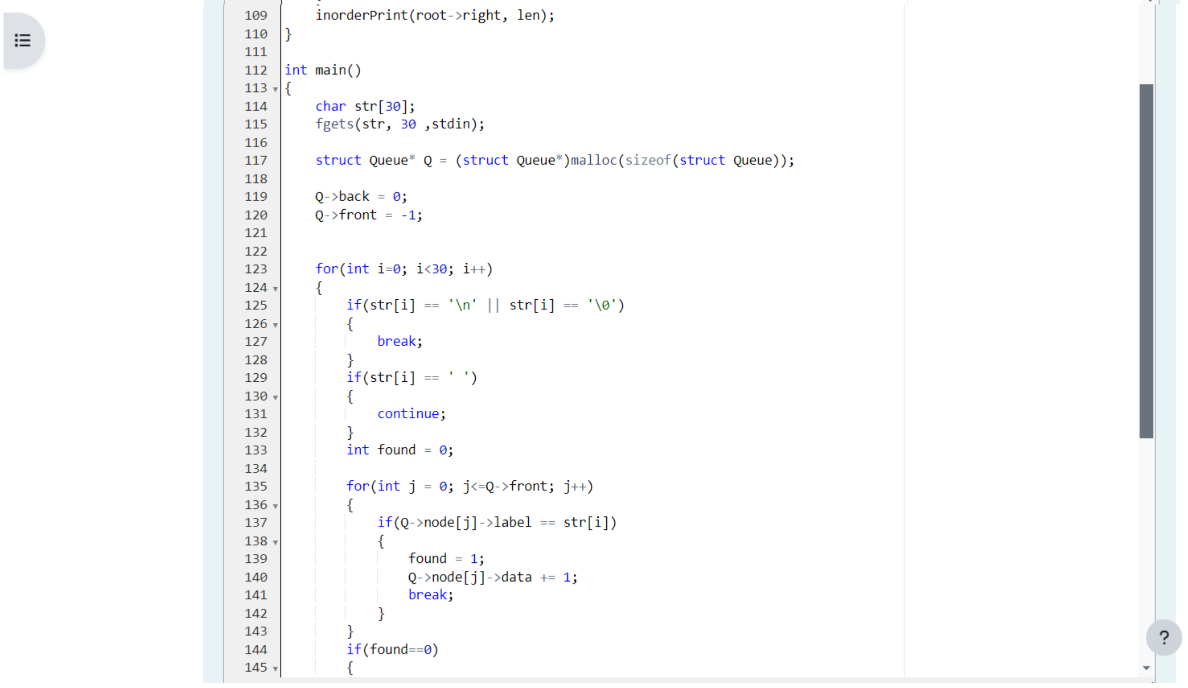
```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct Node {
    int data;
    char label;
    char alph;
    char hc[10];
    struct Node* left;
    struct Node* right;
};

struct Queue
{
    int front;
    int back;
    struct Node* node[30];
};

struct Node* createNode(int data, char label, char alph)
{
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->left = NULL;
    node->right = NULL;
    node->data = data;
    node->label = label;
    node->alph = alph;
    node->hc[0] = '\0';
    return node;
}

void sortQ(struct Queue* Q)
{
    for(int i = Q->back; i<=Q->front; i++)
    {
```



```c
        for(int j=Q->back; j<=Q->front-1; j++)
        {
            if(Q->node[j]->data > Q->node[j+1]->data)
            {
                struct Node* temp = Q->node[j];
                Q->node[j] = Q->node[j+1];
                Q->node[j+1] = temp;
            }
            else if(Q->node[j]->data == Q->node[j+1]->data)
            {
                // printf("    %c - %c\n",Q->node[j]->alph, Q->node[j+1]->alph);
                if(Q->node[j]->alph > Q->node[j+1]->alph)
                {
                    struct Node* temp = Q->node[j];
                    Q->node[j] = Q->node[j+1];
                    Q->node[j+1] = temp;
                }
            }
        }
    }
}

void sortQwithoutLabel(struct Queue* Q)
{
    for(int i = Q->back; i<=Q->front; i++)
    {
        for(int j=Q->back; j<Q->front; j++)
        if(Q->node[j]->data < Q->node[j+1]->data)
        {
            struct Node* temp = Q->node[j];
            Q->node[j] = Q->node[j+1];
            Q->node[j+1] = temp;
        }
    }
}
```



```c

void inorderHuffman(struct Node* root, char str[10], int len)
{
    if(root == NULL)
    {
        return;
    }

    str[len] = '0';
    inorderHuffman(root->left, str, len+1);

    for(int i = 0; i<len; i++)
    {
        root->hc[i] = str[i];
    }
    root->hc[len+1] = '\0';
    // if(root->label != '~')
    // {
    //     printf("%c-",root->label);
    //     puts(root->hc);
    // }
    str[len] = '1';
    inorderHuffman(root->right, str, len+1);
}

void inorderPrint(struct Node* root, int len)
{
    if(root == NULL)
    {
        return;
    }
    inorderPrint(root->left, len);
    if(root->label != '~' && strlen(root->hc) == len)
    {
        printf("%c-",root->label);
        puts(root->hc);
    }
```



```c
    inorderPrint(root->right, len);
}

int main()
{
    char str[30];
    fgets(str, 30 ,stdin);

    struct Queue* Q = (struct Queue*)malloc(sizeof(struct Queue));

    Q->back = 0;
    Q->front = -1;


    for(int i=0; i<30; i++)
    {
        if(str[i] == '\n' || str[i] == '\0')
        {
            break;
        }
        if(str[i] == ' ')
        {
            continue;
        }
        int found = 0;

        for(int j = 0; j<=Q->front; j++)
        {
            if(Q->node[j]->label == str[i])
            {
                found = 1;
                Q->node[j]->data += 1;
                break;
            }
        }
        if(found==0)
        {
            Q->front+=1;
            Q->node[Q->front] = createNode(1,str[i],str[i]);
        }
    }

    sortQwithoutLabel(Q);

    printf("FREQUENCY OF ALPHABETS\n");
    for(int j = Q->back; j<=Q->front; j++)
    {
        printf("%c-%d\n",Q->node[j]->label,Q->node[j]->data);
    }

    while(Q->front - Q->back > 0)
    {
        sortQ(Q);
        struct Node* left = Q->node[Q->back];
        struct Node* right = Q->node[Q->back+1];
        int nd = left->data + right->data;

        Q->node[Q->front+1] = createNode(nd, '~', left->alph);
        Q->node[Q->front+1]->left = left;
        Q->node[Q->front+1]->right = right;

        Q->back+=2;
        Q->front+=1;
    }

    printf("HUFFMAN CODE IS\n");

    char temp[10];
    inorderHuffman(Q->node[Q->front], temp, 0);

    for(int x = 5; x>0; x--)
    {
        inorderPrint(Q->node[Q->front], x);
    }
}
```

	Test	Input	Expected	Got	
✓	1	ADAM IS IN MADAMS KITCHEN	FREQUENCY OF ALPHABETS A-4 M-3 I-3 D-2 S-2 N-2 K-1 T-1 C-1 H-1 E-1 HUFFMAN CODE IS C-11010 E-11011 H-0010 K-0011 T-1100 D-000 N-010 S-011 I-100 M-101 A-111	FREQUENCY OF ALPHABETS A-4 M-3 I-3 D-2 S-2 N-2 K-1 T-1 C-1 H-1 E-1 HUFFMAN CODE IS C-11010 E-11011 H-0010 K-0011 T-1100 D-000 N-010 S-011 I-100 M-101 A-111	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 2.00

Flag question

1. Write a program to compute the Longest Common Subsequence using Dynamic Programming

For example:

Test	Input	Result
1	mother Theresa other	The longest common sub sequence is : other
2	daredevildead devil	The longest common sub sequence is : devil

Answer: (penalty regime: 0 %)

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

int max(int a, int b)
{
    return a>b?a:b;
}

int main()
{
    char s1[20], s2[20];
    fgets(s1, 20, stdin);
    fgets(s2, 20, stdin);

    // strcpy(s1, "stone\n");
    // strcpy(s2, "longest\n");

    int l1 = strlen(s1);
    int l2 = strlen(s2);

    l1--;
    l2--;

    int mat[l1+1][l2+1];

    for(int i = 0; i<l1+1; i++)
    {
        mat[i][0] = 0;
    }
    for(int i = 0; i<l2+1; i++)
    {
        mat[0][i] = 0;
    }

    for(int i = 1; i<l1+1; i++)
    {
        for(int j = 1; j<l2+1; j++)
        {
            if(s1[i-1] == s2[j-1])
            {
                mat[i][j] = mat[i-1][j-1] + 1;
            }
            else
            {
                mat[i][j] = max(mat[i][j-1],mat[i-1][j]);
            }
        }
    }

    char common[20];
    int p = 0;
    int x = l2;
    for(int i = l1; i>0; i--)
    {
        if(mat[i][x] == mat[i-1][x])
        {
            continue;
        }
        common[p++] = s2[--x];
    }

    printf("The longest common sub sequence is : ");

    for(int i = p-1; i>=0; i--)
    {
        printf("%c", common[i]);
    }

    return 0;
}
```

	Test	Input	Expected	Got	
✓	1	mother Theresa other	The longest common sub sequence is : other	The longest common sub sequence is : other	✓
✓	2	daredevildead devil	The longest common sub sequence is : devil	The longest common sub sequence is : devil	✓

Passed all tests! ✓

Question 4

Correct

Marked out of 2.00

Flag question

Write a program to find the maximum sub array sum using the given array {-2, -5, 6, -2, -3, 1, 5, -6}

For example:

Test	Input	Result
1	8 -2 -5 6 -2 -3 1 5 -6	The maximum sub array is [6 -2 -3 1 5] The maximum sub array sum is :7
2	7 -4 5 7 -6 10 -15 3	The maximum sub array is [5 7 -6 10] The maximum sub array sum is :16

Answer: (penalty regime: 0 %)

```
#include<stdio.h>

int subArrSum(int* arr, int beg, int end, int* first,int* last)
{
    if(beg == end)
    {
        return arr[beg];
    }

    int mid = (beg+end)/2;

    int ls = subArrSum(arr, beg, mid, first, last);
    int rs = subArrSum(arr, mid+1, end, first, last);

    int lss = arr[mid];
    int rss = arr[mid+1];

    int l = mid, r = mid+1;
    for(int i = beg; i<= mid -1; i++)
    {
        int tlss = 0;
        for(int j = i; j<=mid; j++)
        {
            tlss+=arr[j];
        }
        // lss = tlss>lss?tlss:lss;
        if(tlss>lss)
        {
            lss = tlss;
            l = i;
        }
    }

    for(int i = mid+2; i<=end; i++)
    {
        int trss = 0;
        for(int j = mid+1; j<=i; j++)
        {
            trss+=arr[j];
        }
        if(trss>rss)
        {
            rss = trss;
            r = i;
        }
    }

    int cs = lss+rss;
    int max = 0;

    if(cs>ls && cs>rs)
    {
        max = cs;
        *first = l;
        *last = r;
    }

    return max;
}

int main()
{
    int n;
    scanf("%d", &n);
    int a[n];

    for(int i=0; i<n;i++)
    {
        scanf("%d", &a[i]);
    }

    int first = a[n/2], last = a[n/2 + 1];
    int ans = subArrSum(a, 0, n-1, &first, &last);

    printf("The maximum sub array is [%d", a[first]);
    for(int x = first+1; x<=last; x++)
    {
        printf(" %d", a[x]);
    }
    printf("]\nThe maximum sub array sum is :");
    printf("%d", ans);

    return 0;
}
```

	Test	Input	Expected	Got	
✓	1	8 -2 -5 6 -2 -3 1 5 -6	The maximum sub array is [6 -2 -3 1 5] The maximum sub array sum is :7	The maximum sub array is [6 -2 -3 1 5] The maximum sub array sum is :7	✓
✓	2	7 -4 5 7 -6 10 -15 3	The maximum sub array is [5 7 -6 10] The maximum sub array sum is :16	The maximum sub array is [5 7 -6 10] The maximum sub array sum is :16	✓

Passed all tests! ✓

Question 5
Correct
Marked out of 2.00
Flag question

write a program to implement the matrix chain multiplication and compute the number of multiplications needed by applying Dynamic Programming Approach.

Consider the problem of multiplying 4 matrices A1,A2,A3,A4

note : read the number of matrices to be used

read the domain of all the matrices

display the output with total number of multiplication needed to solve the problem.

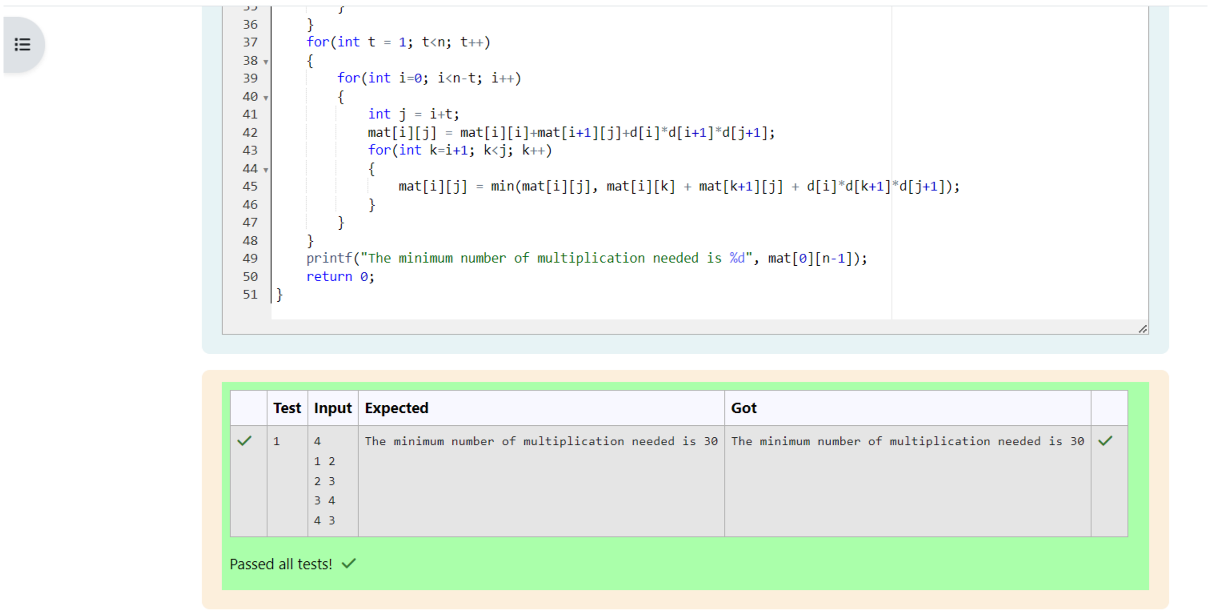
For example:

Test	Input	Result

1	4
	1 2
	2 3
	3 4
	4 3
		The minimum number of multiplication needed is 30

Answer: (penalty regime: 0 %)

```
#include<stdio.h>

int min(int a, int b)
{
    return a<b?a:b;
}

int main()
{
    int n;
    scanf("%d", &n);

    int d[n+1];
    int x;

    scanf("%d", &d[0]);
    for(int i=1; i<n; i++)
    {
        scanf("%d %d", &x, &d[i]);
        if(x!=d[i])
        {
            printf("*%d %d Error. Invalid input.*", x, d[i]);
            return 1;
        }
    }
    scanf("%d",&d[n]);

    int mat[n][n];

    for(int i=0;i<n;i++)
    {
        for(int j=0;j<=i;j++)
        {
            mat[i][j] = 0;
        }
```



```c
        }
    }
    for(int t = 1; t<n; t++)
    {
        for(int i=0; i<n-t; i++)
        {
            int j = i+t;
            mat[i][j] = mat[i][i]+mat[i+1][j]+d[i]*d[i+1]*d[j+1];
            for(int k=i+1; k<j; k++)
            {
                mat[i][j] = min(mat[i][j], mat[i][k] + mat[k+1][j] + d[i]*d[k+1]*d[j+1]);
            }
        }
    }
    printf("The minimum number of multiplication needed is %d", mat[0][n-1]);
    return 0;
}
```

	Test	Input	Expected	Got	
✓	1	4 1 2 2 3 3 4 4 3	The minimum number of multiplication needed is 30	The minimum number of multiplication needed is 30	✓

Passed all tests! ✓