Name : Shubham Dutta

Section : 2B

Year : 2nd

Stream : CST

Roll : 58

Enrollment : 12019009022112

Paper Name : Design And Analysis of Algorithm

Paper Code : PCC - CS492

Date : 21/05/2021

Time : 9:30 - 12:30 AM

Signature : Shubham Dutta

Answer

1 (a)

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
void swap (int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}
int partition (int array[], int low, int high) {
    int pivot = array[high];
    int i = (low - 1);
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            i++;
            swap (&array[i], &array[j]);
        }
    }
    swap (&array[i+1], &array[high]);
    return (i+1);
}
```

```
void printArray(int array[], int size) {
  for (int i = 0; i < size; i++) {
    printf("%d ", array[i]);
  }
  printf("\n");
}
int main() {
  int data[] = {4, 3, 1, 5, 2};
  int n = sizeof(data) / sizeof(data[0]);
  quickSort(data, 0, n-1);
  printArray(data, n);
  return 0;
}
```

---

2 Ans.

```
N, M = input().split()
N = int(N)
M = int(M)
G = [[] for _ in range(N+1)]
exist = set()
for _ in range(M):
    a, b = input().split()
    a = int(a)
    b = int(b)
```

```
if (a,b) not in exist and (b,a) not in exist:
        G[a].append(b)
        G[b].append(a)
        exist.add((a,b))
x = int(input())
visited = [0 for _ in range(N+1)]
def DFS(G, source):
    visited[source] = 1
    for v in G[source]:
        if visited[v] == 0:
            DFS(G, v)
DFS(G, x)
ans = visited.count(0) - 1
print(ans)
```

---

3 Ans:

```
class node:
    def __init__(self, freq, symbol, left=None,
                 right=None):
        self.freq = freq
        self.symbol = symbol
        self.left = left
        self.right = right
        self.huff = 0,
```

```
def printNodes(node, val=''):
    newVal = val + str(node.huff)
    if(node.left):
        printNodes(node.left, newVal)
    if(node.right):
        printNodes(node.right, newVal)
    if(not node.left and not node.right):
        mama.append([node.symbol, int(newVal)])
mama = []
z = int(input())
chars = []
chars = input().split()
freq = list(map(int, input().split()))
nodes = []
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))
while len(nodes) > 1:
    nodes = sorted(nodes, key = lambda x: x.freq)
    left = nodes[0]
    right = nodes[1]
    left.huff = 0
    right.huff = 1
    newNode = node(left.freq + right.freq, right.symbol
                   + left.symbol, left, right)
```

```
        nodes.remove(left)
        nodes.remove(right)
        nodes.append(newNode)
    printNodes(nodes[0])
```

~~for~~

```
for j in range(len(chars)):
    for i in range(0, z):
        if(chars[j] == mama[i][0]):
            print("{} = {}".format(chars[i],
                                   mama[i][j]))
            break.
```

Q4 ans

```
def MatrixChainOrder(p, n):
    dp = [[0 for i in range(n)]
          for i in range(n)]
    for i in range(1, n):
        dp[i][j] = 0
    for L in range(1, n-1):
        for i in range(n-L):
            dp[i][i+L] = min(dp[i+1][i+L]+
                p[i-1]*p[i]*p[i+L],
                dp[i][i+L-1] +
                p[i-1]*p[i+L-1]*p[i+L])
```

```
        return dp[1][n-1]
n = int(input())
arr = []
for i in range(0, n):
    a = list(map(int, input().split()))
    arr.append(a[0])
arr.append(a[1])
size = len(arr)
output = MatrixChainOrder(arr, size)
if(output == 8):
    print(output-1)
else:
    print(output)
```