Programming Questions

1. Create a function *mergeStrings* function that takes in two parameters “a” and “b”. Your function must merge strings “a” and “b”, and then return a single merged string. A merge operation on two strings is described as follows:

   - Append alternating characters from “a” and “b”, respectively, to some new string, *mergedString*.
   - Once all of the characters in one of the strings have been merged, append the remaining characters in the other string to *mergedString*.

   Input Format
   The two strings, “a” and “b”, through stdin and passes them to your function.

   Constraints
   - $1 \leq |a|, |b| \leq 25000$

   Output Format
   Your function must return the merged string. This should be printed to stdout in your code.

Sample Input 1
abc def
Sample Output 1
Adbecf
Sample Input 2
ab zsd
Sample Output 2
azbsd

   Explanation
   Sample Case 1
   a = abc
   b = def
   Taking alternate characters from both the strings, we get *adbecf*

   Sample Case 2
   a = ab
   b = zsd
   Taking alternate characters from both the strings, we get *azbsd*

2. Create a *removeNodes* function provided in your editor. It has two parameters:

- list: A reference to a LinkedListNode that is the head of a linked list.
- x: An integer value.

Your function should remove all nodes from the list having data values greater than x, and then return the head of the modified linked list.

Input Format
The locked stub code in your editor processes the following inputs and passes the necessary arguments to the removeNodes function: The first line contains N , the number of nodes in the linked list. Each line i (where $0 \leq i < N$) of the N subsequent lines contains an integer representing the value of a node in the linked list. The last line contains an integer, x.

Constraints
- $1 \leq N, x \leq 10^5$
- $1 \leq list_i \leq 10^5$, where $0 \leq i < N$

Output Format
Return the linked list after removing the nodes containing $values > x$.

Sample Input 1
5 1 2 3 4 5 3
Sample Output 1
1 2 3
Sample Input 2
5 5 2 1 6 7 5
Sample Output 2
5 2 1

Explanation

Sample Case 1:

$N = 5, x = 3$
$list = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$
After removing the nodes having $value > 3$, $list = 1 \rightarrow 2 \rightarrow 3$

Sample Case 2:

$N = 5, x = 5$
$list = 5 \rightarrow 2 \rightarrow 1 \rightarrow 6 \rightarrow 7$
After removing the nodes having $value > 5$, $list = 5 \rightarrow 2 \rightarrow 1$

3. A mechanical engineer is writing a design specification for two gears to transmit motion between two parts, A and B, in a machine she is designing. The distance between A and B is equal to D. There are n types of gears. A gear of type i has a radius R and cost C . The two gear types specified, i and j, must have $R_i + R_j \geq D$, in order for there to be a way of placing them so that they touch and work together. The objective is to find the pair which costs the least. You need to produce a design table that gives the most suitable match for every gear type in the list. For every gear type i, you need to consider its description (R , C) and list the gear type j to pair with i in table position T . The best match might be the same type ($T = i$). If there are multiple solutions with the same cost, choose the gear with the largest radius. If both the cost and radius you need are found in more than one gear type, choose the type with the smallest index j. If no gears can be found that allow the distance D to be covered, the table should contain 0.

Constraints
All values are integers.
$n \in [1, 10]$
$i, j, T \in [1, n]$
$D, R, C \in [1, 10]$

Input
$n\ D$
$R_1\ R_2\ \ldots\ R_n$
$C_1\ C_2\ \ldots\ C_n$

Output
$T_1\ T_2\ \ldots\ T_n$

Sample Input
5 8
1 3 6 2 5
5 6 8 3 4

Sample Output
0 5 4 3 5

Explanation
For $type$ 1, R is 1 so no gear in the list meets the requirement. Output 0.

For $type$ 2, R is 3, so $type$ 3: $(6, 8)$ and $type$ 5: $(5, 4)$ meet the radius requirement, but 5: $(5, 4)$ has the lower cost, so choice T is 5.

For $type$ 3, R is 6, so the set of candidates is $\{2: (3, 6), 3: (6, 8), 4: (2, 3), 5: (5, 4)\}$. 24: $(2, 3)$ costs the least, so choice T is 4.

For $type$ 4, R is 2, only 3: (6, 8) meets the radius requirement, so choice T is 3.

For $type$ 5, R is 5, so $\{2: (3, 6), 3: (6, 8), 5: (5, 4)\}$ is the set of candidates, and 5: $(5, 4)$ costs the least, so choice T is 5.

4. Would the code compile fine? If not, what change needs to be done?

```
template <typename T>
class Foo{
T tVar;
public:
Foo(T t) : tVar(t) { }
};
class FooDerived : public Foo<std::string> {};
int main(){
FooDerived d;
return 0;
}
```

a) The code would compile without any errors

b) Compiler Error; FooDerived is a non-template class that derives from a template class, it should be changed

c) Compiler error; tVar is a variable of an unknown type

a) Compiler error; It can be fixed by adding an empty constructor to Foo class

5. Predict the output

```
class Someclass {
public:
int x;
public :
Someclass(int xx) : x(xx) { }
Someclass(const Someclass& a) { x = a.x ; x++;}
void operator =(const Someclass& a1) { x = a1.x ; x--;}
};
int main( )
{
Someclass a(4);
Someclass b = a;
cout << b.x << endl;
return 0;
}
```

a) 4

b) 3

c) 5

d) Compile error

6. How do u declare a pointer to an array of pointers to int?

a) int *a[5];

b) int **a[5];

c) int *(*a)[5];

a) None of these