```
#include <iostream>
#include <cassert>
Q2 template <class T>
class Stack
{
private:
    T* arr;
    const int size;
    int top;
public:
    Stack(int size = 1000) : size{size} {}
    {
    assert (size > 0);
    arr = new T[size];
    top = -1;
    }
  Stack <T>* Push(const T& val)
  {
    top++;
    assert ( top < size);
    arr[top] = val;
    return this;
  }
  void Pop()
  {
    assert ( top >= 0);
    top--;
  }
```

```
void Increment (int K , T val)
{
    assert ( K <= size);
    for (int i=0 ; i<k; i++)
    {
        if ( i > top)
            break;
        arr [i]  += val;
    }
}
```

~~const void Top() const~~

```
};
```

Q3

Pseudo - code :-

```
stack <string> stack ;
queue <string> end ;
string middle;

// Take length
int n;
Cin >> n;
// Make char array, store values.
char arr[n];
// Input every char into the array.

// ① Find MIDDLE.

for( int i=0; i<n; i++)
  {
    if(arr[i] != '#' && arr[i] != '$')
                middle += arr[i];
    else
      break;
  }.
```

② Find START. `i = 0`

```
while (i < n)
{ if (arr[i] == '#')
  {
    string temp;
    i++;
    while( i<n && arr[i] != '$' &&
                  arr[i] != '#' )
    { temp += arr[i];
      i++;
    }
    start.push(temp);
  }
  else {i++;}.
}.
```

// Now start is a stack containing all starting characters.

③ Do the same with ~~cont~~ END. However, END will be a Queue.

④ To print answer :-
Output ~~through~~ strings in stack until stack is empty.
Output middle
Output queue.front() until queue is empty.

Q4 If $z_2 = q$.

Then :- $S[1] = S[2]$

$S[2] = S[3]$

$S[3] = S[4]$

⋮

$S[q] = S[q+1]$.

$S[q+1] \neq S[q+2]$.

That is, the first $(q+1)$ letters of S are same.
and the $(q+2)$ letter is different.

So, $z_3 = q-1$, $z_4 = q-2$,

....

$z_q = 2$, $z_{q+1} = 1$ ($S[q+1] = S[1]$)

and $z_{q+2} = 0$ ($S[q+2] \neq S[1]$).

Q6 Find Z values for B by comparing with ~~prefix~~ ~~suffix~~ prefix of A.

To find Z-values of B, the only difference is that for Z_i, instead of comparing B[i...] with B[1...] we will compare B[i...] with A[1...].

This can be done in O(n) time.

Finally, we go through all Z values of B and find the value i such that $i + Z[i] - 1 = len(B)$

where $len(B)$ = length of B.

The smallest value of i such that $i + Z[i] - 1 = len(B)$.

The answer will be Z[i].

Example :-

B = a b
Z[i] = 0 0 . . . . .
(One indexing) 1 2

A → (e f a b d i j)

B =	a	b	c	d	e	f	a	b	d
Z[i] =	0	0	0	0	5	0	0	0	0
i =	1	2	3	4	5	6	7	8	9

(1-indexing)

$len(B) = 9$.

For $i = 5$, $i + Z[i] - 1 = 5 + 5 - 1 = 9 = len(B)$.

→ Ans. = Z[5] = 5.