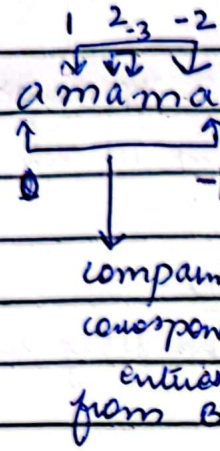
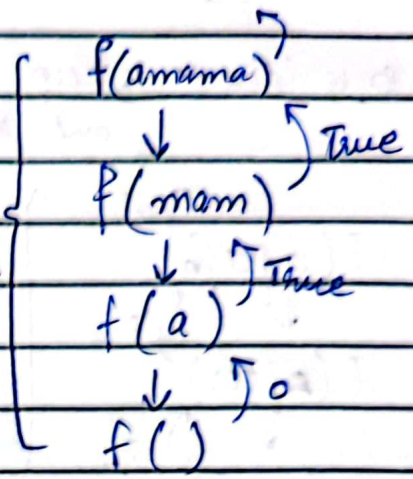
Q1 PALINDROME



comparing corresponding entries from B.S. → can be achieved by removing first and last character at each call/step.

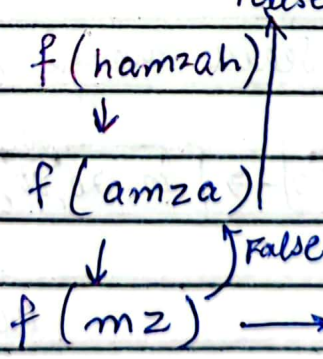
First and last character are same



Base condition. return 0 when string is empty (length=0)

Recursive call: function called recursively with the substring excluding first and last characters.

Neg. case. If first and last characters are not same, recursive call will stop and function will not call itself anymore + return False (not palindrome).



→ not same first and last letters so function returns False and exits.

Q2

REVERSE A STRING.

Base idea → : Keep reducing the size of string until its length is 0 and then print first character of substring at each step.

f(after) → 'a'
f(after) → 'f'
f(after) → 't'
f(ter) → 'e'
f(er) → 'r'
f(r) → 0
f() Base cond.

character at first index is printed at each call.

Base cond.
return 0 when string empty

Recursive case.
Recursively call function with substring excluding first character.

Q3

Binary Search through recursion.

key 20

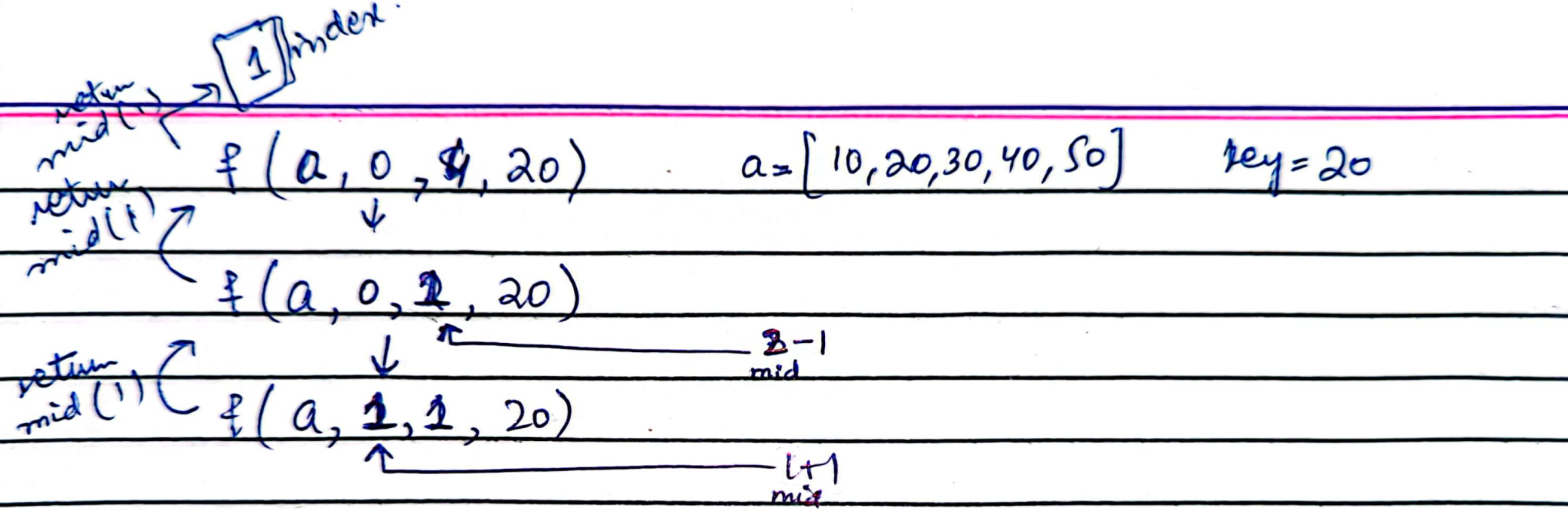
[10, 20, 30, 40, 50] → [10, 20, 30, 40, 50] → [10, 20, 30, 40, 50] answer.
0 1 2 3 4

mid

[10, 20]
mid = 15 (<20)
[20]
mid — return index

Base condition.
return -1 when start > end because that's when the list is search as much as possible to find the key.

Recursive calls.
call the function with changed values of start/end to speed up search as done in binary search.


1
index
return mid(1)
f(a, 0, 4, 20)
a=[10,20,30,40,50]
key=20
return mid(1)
f(a, 0, 2, 20)
2-1
mid
return mid(1)
f(a, 1, 1, 20)
l+1
mid