Name: Santhosh Kumar N
Roll No: 2022MCS120009

1) The code optimization is a program transformation technique which tries to improve the intermediate code by making it consume fewer resources (cpu, memory).
There are different types of code optimizations.
a) Machine Independent optimizations is one of them code optimizations can be done different ways
1) Compile time evaluation
2) Variable Propagation.
3) Constant Propagation
4) Constant Folding

In above the fictional question. We provided info that improving the processing power of the Computer 10 folds will reduce the execution time of the algorithm to around 1 second.
Redundant computations or unnecessary computations result in inefficiency in the implementations of the algorithms,
when Redundant calculations are embedded inside the loop for the variable which remains unchanged throughout the entire execution phase of the loop.
The results are more serious
Ex: Consider following code in which the value a*a*a is redundantly calculated in the loop.

```
x=0;
for i=0 to n
    x=x+1
    y=(a+a+a*c)*x+x+5+b*b*x;
    Print x, y
next i.
```

Redundant calculations can be removed by small modifications in the program.

2)

(a) [56+29{a-b(m+n)*c}+89]

Here the Balanced expression is

[	56	+	29	{	a	-	b	(	m	+	n	)	*	c	}	+	8	9	]

[56 + 29

(? → No → if yes Push into the stack

)? → No → if yes Pop out of the Stack

Expression	Parantheses balanced () or {} or []
(m+n)	Yes
{(a-b(m+n)*c}	No
{56+29}	No
{a-b*c)	No

Stack:

Str	[	{	(	)	y	]

Push into stack

Step 1:

Stack: [[ ]

Str: [{ [( ) }]

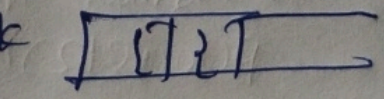
opening stack, Push into stack

Step 2: Stack: [{]

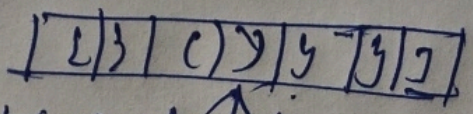
Str: [{ ( ) }]

opening Bracket Push into the Stack

Step 3: Stack



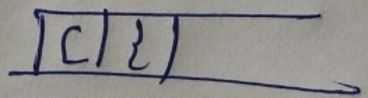
Str



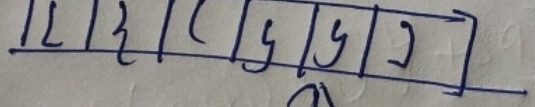
closing bracket. ↑ Check top of stack is same kind or not

Step 4:

Stack



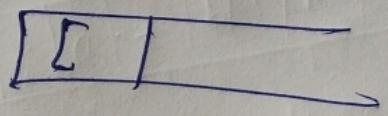
Str



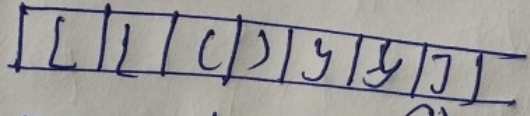
closing bracket. Check the top of stack is same kind or not

Step 5:

Stack



Str



Closing bracket. check top of stack is same kind or not.

(b)

1) Doubly Linked List can be traversed in both forward and backward directions.
2) Deletion of nodes is easier compared to a Single Linked List. because we need access to the node to be deleted and to node previous to it.
3) Reversing a doubly linked list is easy than Single Linked List. Here we only have to swap each node's next and previous pointer and update the head node to point to the last node.

4) Doubly linked list is used in web browsers to implement the backward and forward navigation of web pages through the back and forward buttons.

5) For any given pointer to a particular node in a doubly linked list we can delete the node without traversing any part of the list.

6) Memory space and efficiency is more. & more memory space utilization than singly linked list.