Lab Final Exam

Subject : Advanced Algorithm Lab

Code : CSE 2032

Name : Saleh Ibne Omar

ID : 2017000000040

Section : 01

Semester : Spring - 22

Batch : 45th

Date : June 21, 2022

Time : 11:00 AM

Answer to the Question NO. 04

Text in mynicknameinABC

where My Nick Name in Salen

So, the updated text will be : mynicknameinsalen

all lowercase

BWT :

(Rotation)

mynicknameinsalen $

$mynicknameinsalen

n$mynicknameinsale

en$mynicknameinsal

len$mynicknameinsa

alen$mynicknameins

salen$mynicknamein

ssalen$mynicknamei

issalen$mynickname

eissalen$mynicknam

meissalen$mynickna

ameissalen$mynickn

nameinsalen$mynick

knameinsalen$mynic

eknameinsalen$myni

icknameissalen$myn

nicknameissalen$my

ynicknameissalen$m

Now we have to sort the rotated matrix in lexi. order

Sorted:

$mynicknameinnaleh
aleh$mynicknameinn
ameinnaleh$mynickn
eknameinsaleh$myni
eh$mynicknameinnal
eissaleh$mynicknam
h$mynicknameinsale
icknameinnaleh$myn
innaleh$mynickname
knameinsaleh$mynic
leh$mynicknameinna
meissaleh$mynickna
mynicknameinnaleh$
nameinnaleh$mynick
nicknameinnaleh$my
saleh$mynicknamein
~~naleh~~

ssaleh$mynicknamei
ynicknameinnaleh$m

∴ BWT of the given text is

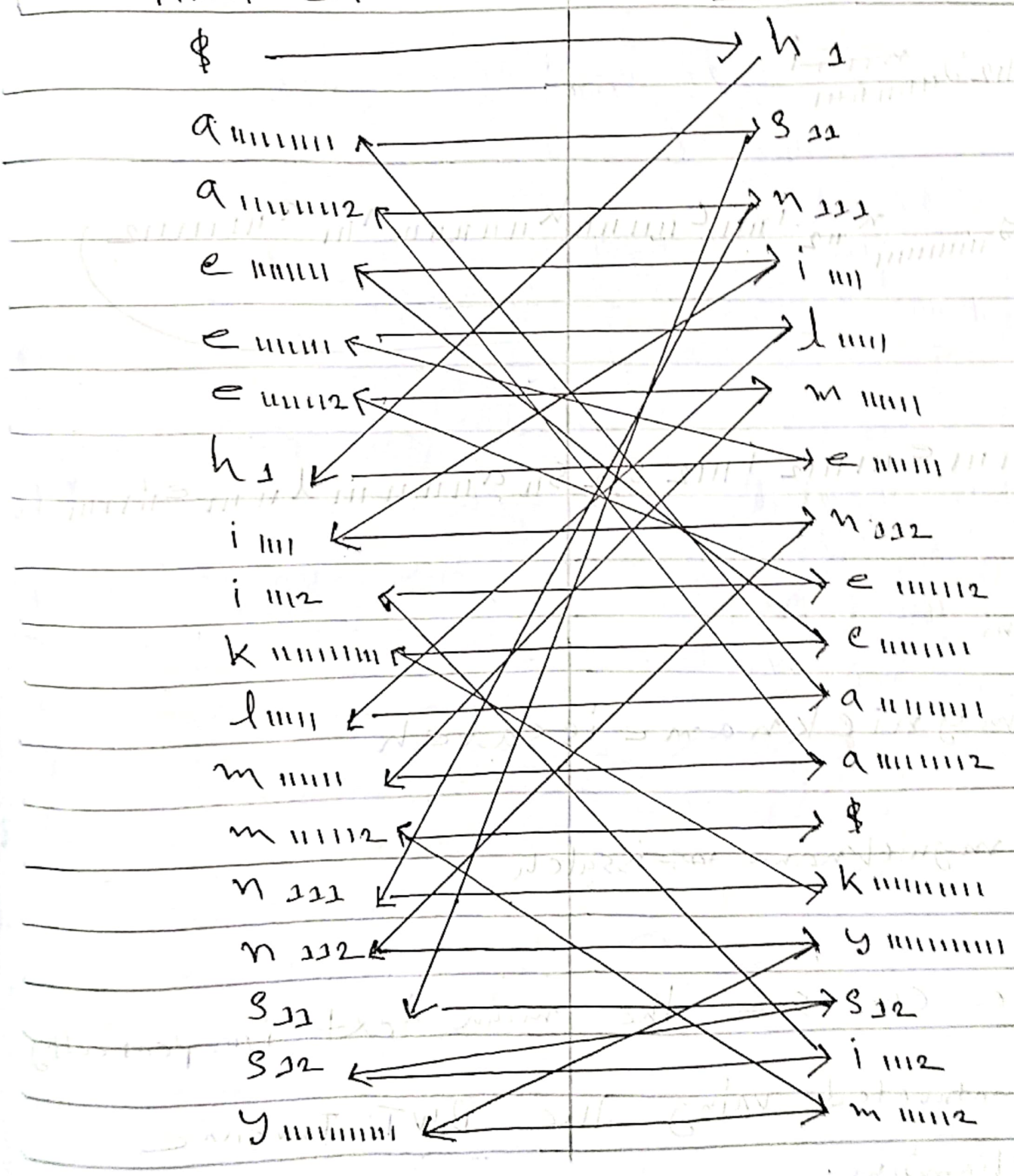
= ~~hsi hsnilmeneeaa$~~

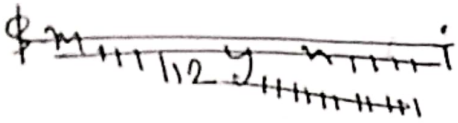
=hsnilmeneeaa$kysim

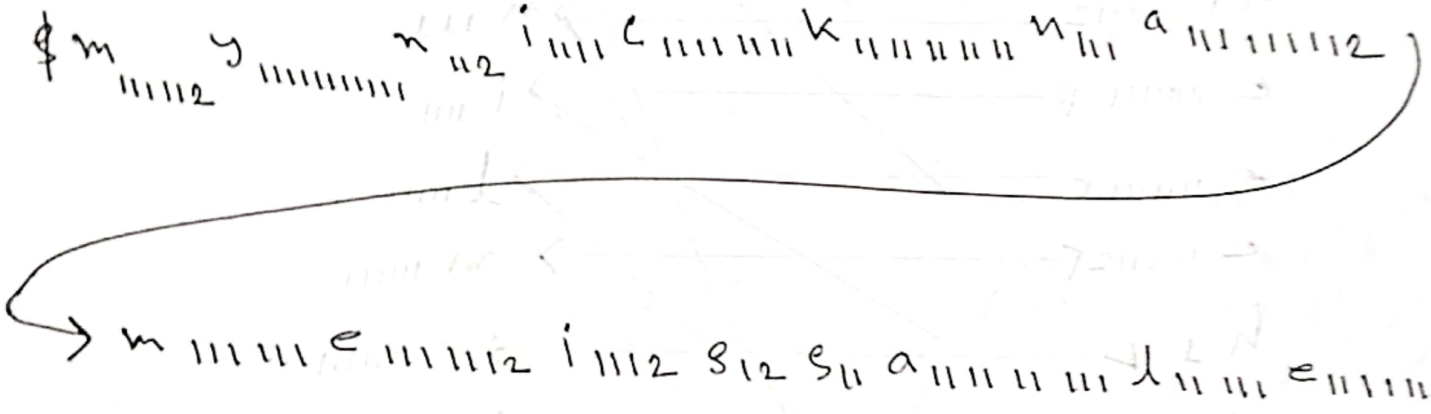
Ans:

P.T.O

Inverting BWT using LF mapping :







~~is $myni~~

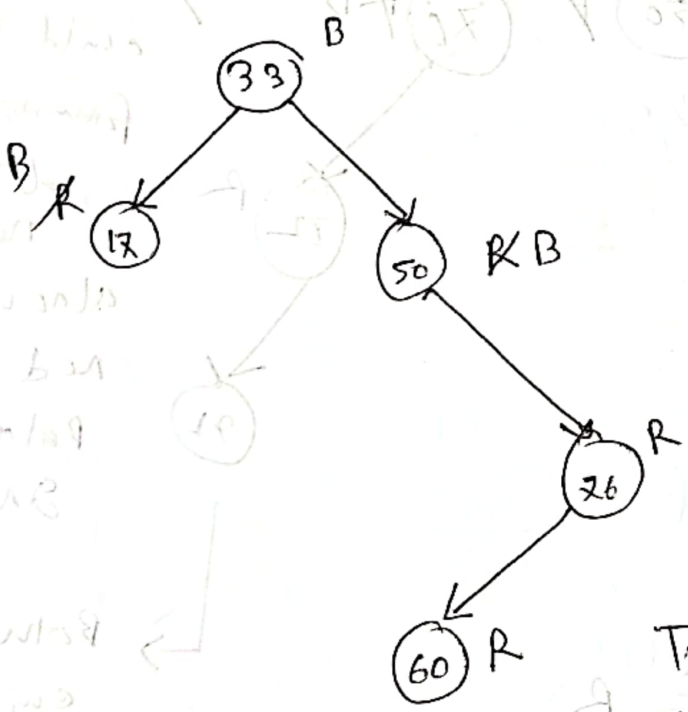
∴ $myniecknameissaleh

⇒ mynieknameissaleh

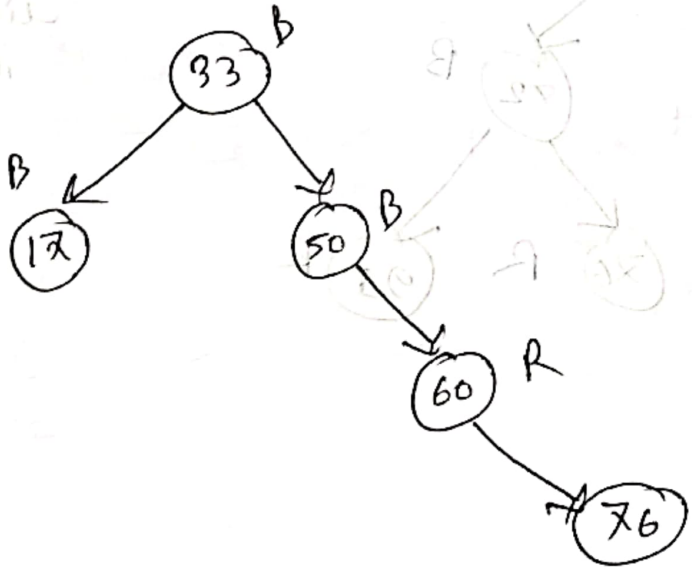
As we can see the main text is perfectly reconstructed using the BWT Inverse mechanism.

Ans: ——————

Answer to the Question No. 05



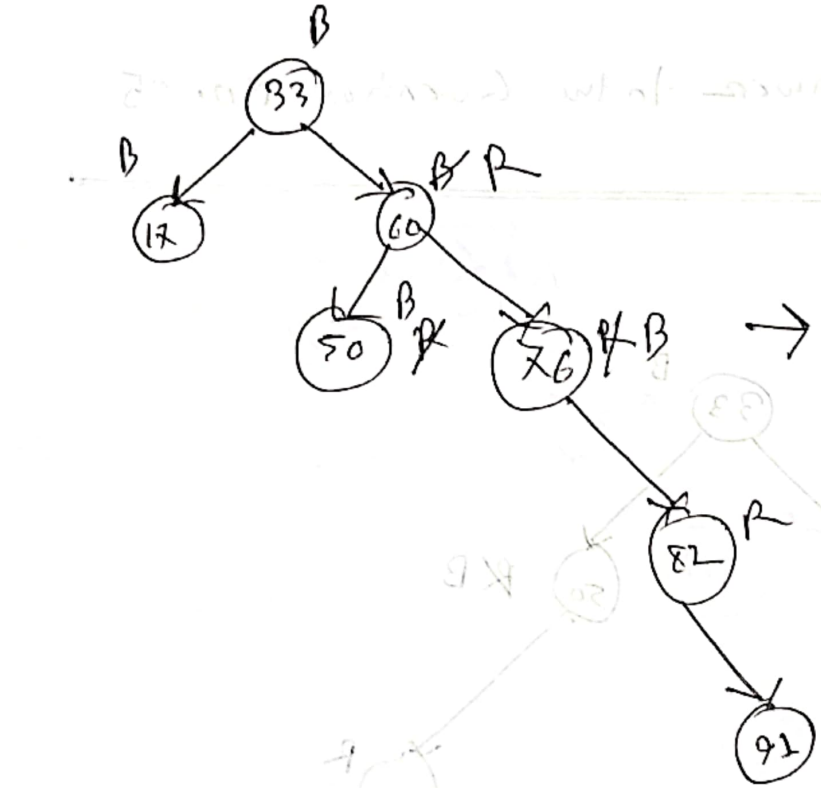
Triange shape both child and parent is red right rotation needed



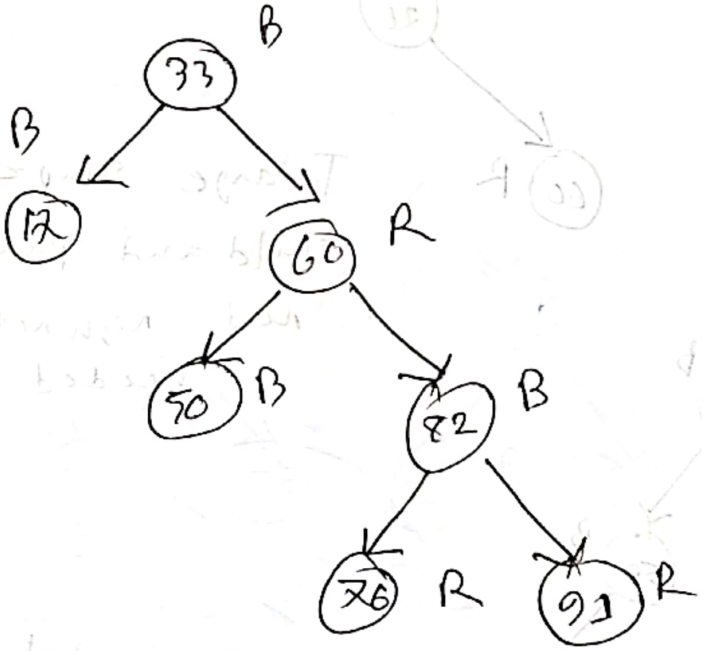
again both parent and child is red and forms a straight line left rotation is needed

P.t.o

2017000000040

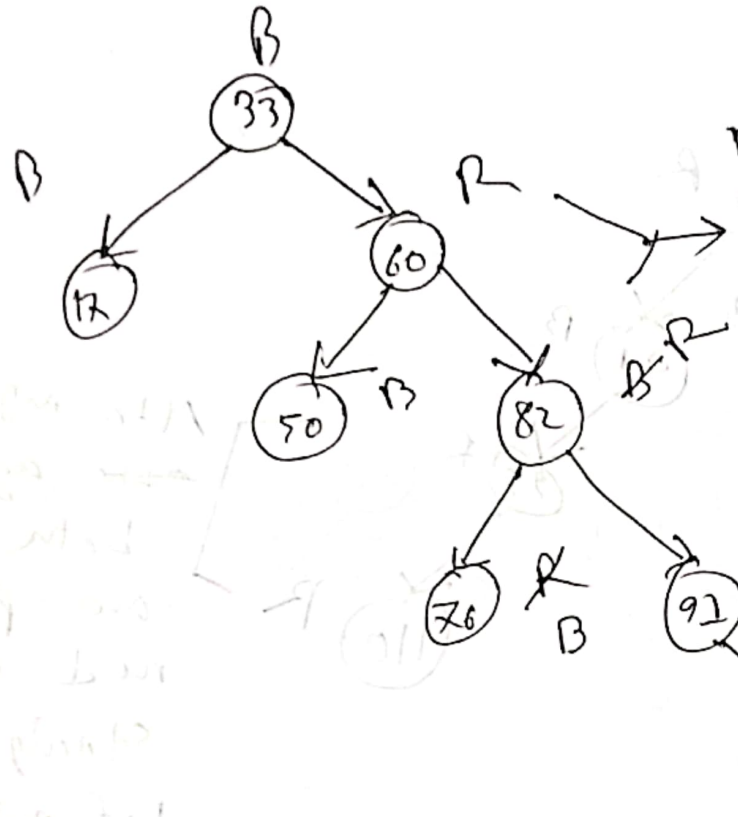


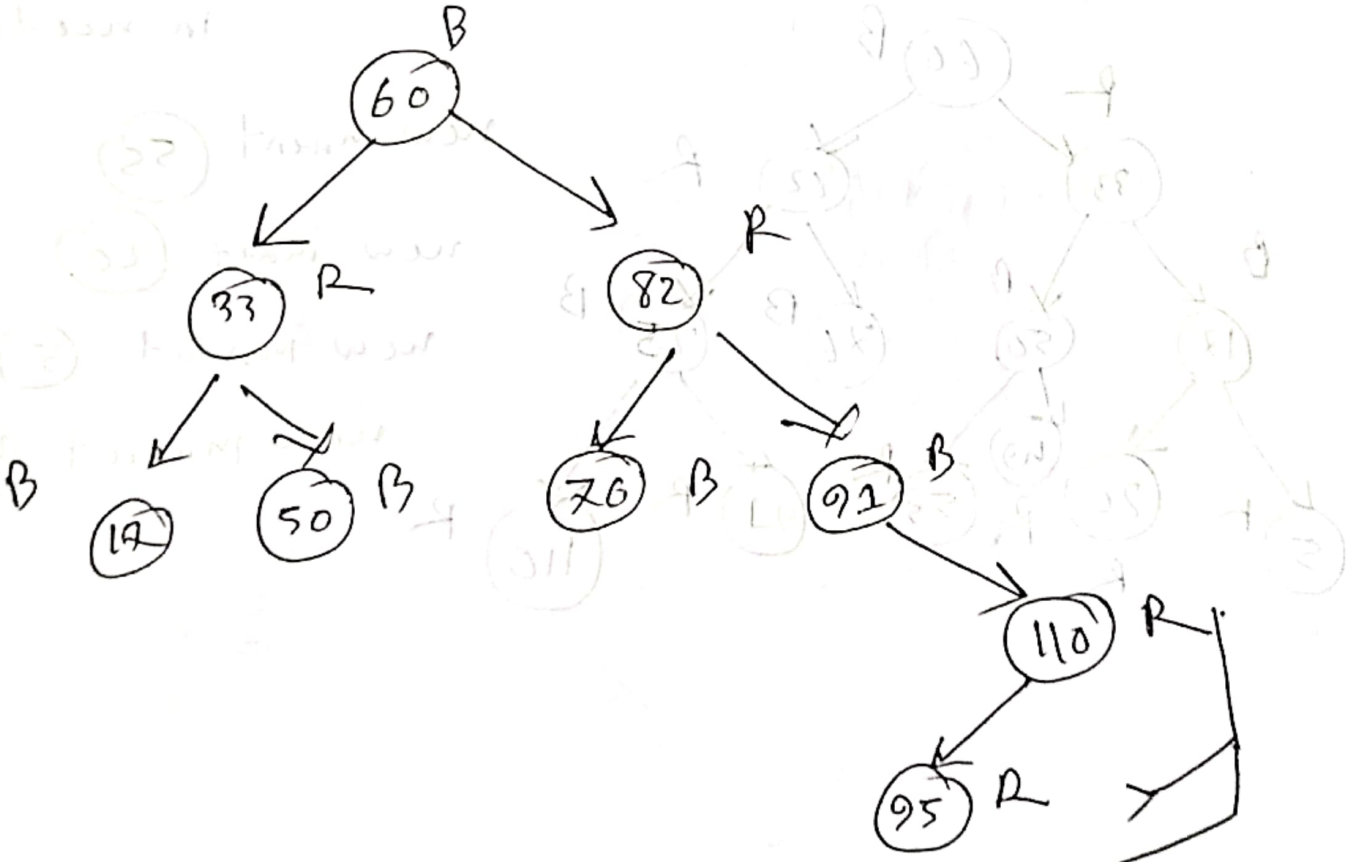
→ Both parent and child in red and ~~form a straight line, left rotation needed~~ also node in red need to balance from grandparent



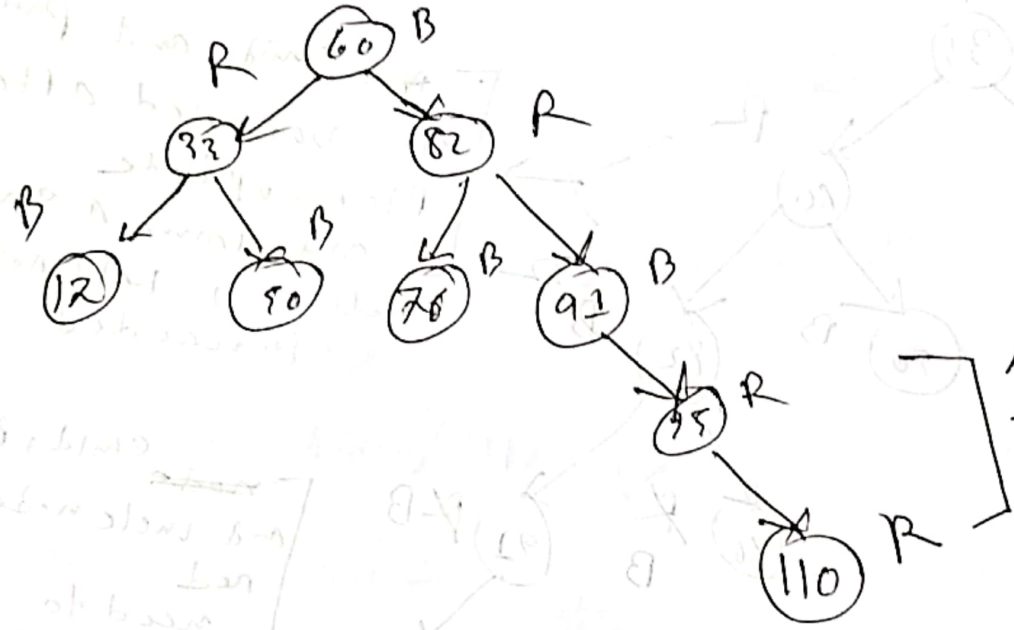
→ Both parent and child in red and forms a straight line left rotation needed

P.T.O

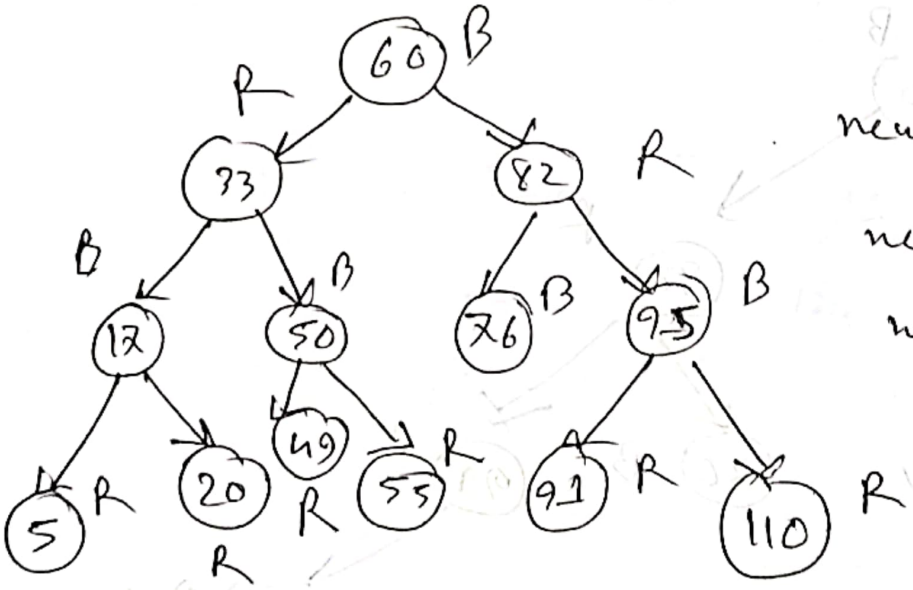

B
33
B
R
17
60
50
B
82
B
R
76
R
B
91
B
R
110
R
~~Art~~ child and parent are both red after Black update and forms a straight line, left rotation is needed
~~node~~ child, parent and uncle node is red need to black from grand parent


B
60
R
33
R
82
B
17
B
50
B
76
B
91
B
110
R
95
R
Both child and parent is red and form triangular shape, right rotation needed

P.t.o



After right rotation ~~again~~ again both child and parent in red and in straight line, left rotation is needed.



new inserted 55

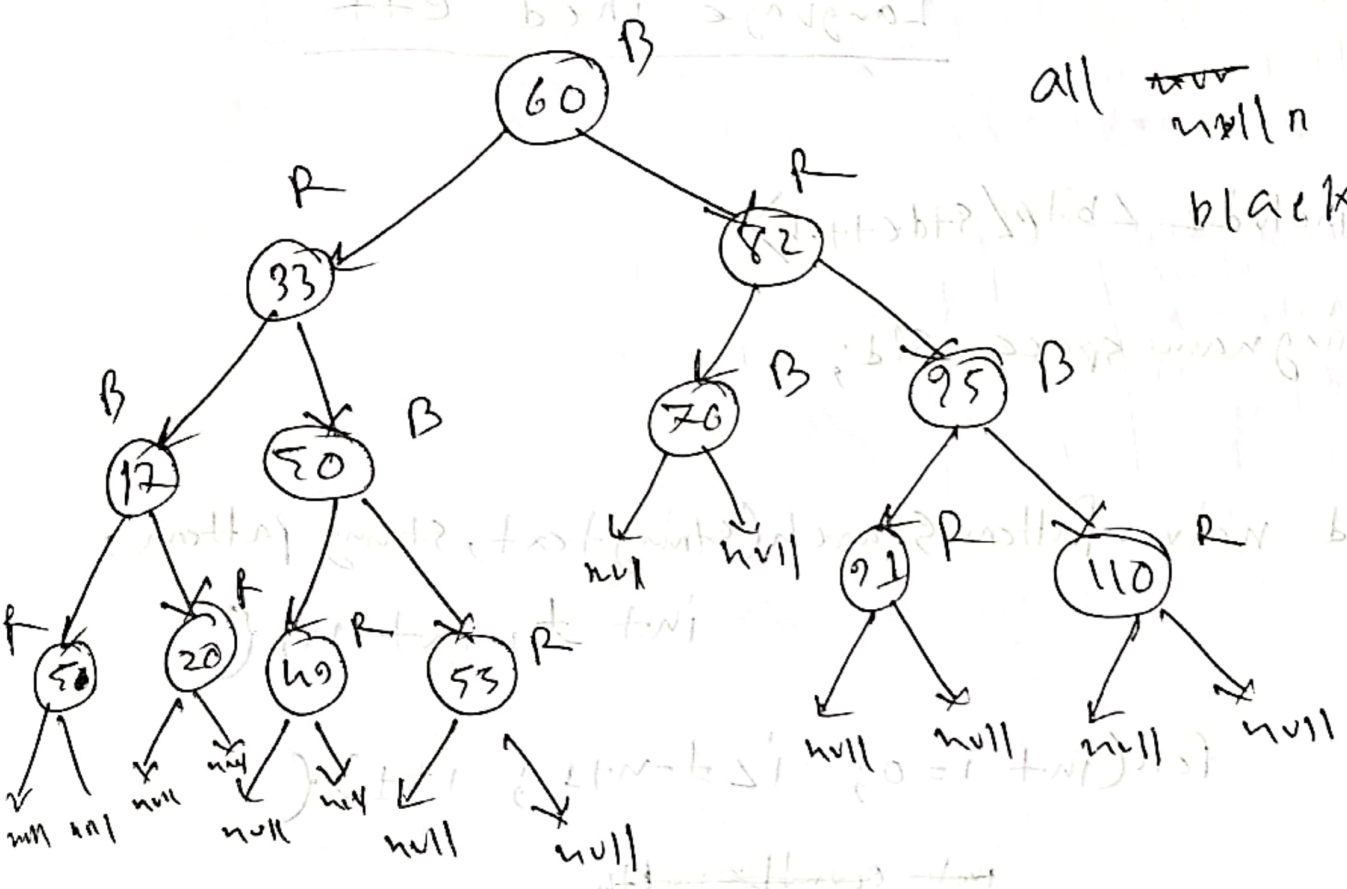
new inserted 20

new inserted 5

new inserted 49

P.T.O

Final RBT is



all null are black

Black heights:

bh(33) = 2

bh(17) = 1

bh(50) = 1

bh(76) = 1

bh(60) = 2

bh(82) = 2

bh(91) = 1

bh(110) = 1

bh(55) = 1

bh(20) = 1

bh(5) = 1

bh(40) = 1

Ans.

Answer to the Question no. 03

Language used C++

```
# include <bits/stdc++.h>

using namespace std;

void naivePatternSearch(string text, string pattern,
                        int t, int n) {

    for(int i=0; i<t-n+1; i++){

        ~~int countLength~~
        int countMatchedLength = 0;

        for(int j=0; j<n; j++){
            if (text[i+j] == pattern[j]) {
                countMatchedLength++;
            }
            else{
                break;
            }
        }
```

P.T.O

```
if(countMatchedLength == n){
    cout << "Pattern found at position " << i << endl;
  }
  else{
    cout << " Pattern doesn't match in position ".
        << i << endl;
  }
}

int main(){
  string text, pattern;
  getline(cin, text);
  cin >> pattern;
 //length of text
int t = text.length();
 //length of pattern
  int n = pattern.length();
```

```
naivePatternSearch(text, pattern, t, n);
~~return 0;~~
return 0;
}
```

Ans: _____

Answer to the Question NO: 01

C++

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int i, j, n, e, VAL;
    cout << "Enter the value of n:" << endl;
    cin >> n;
    cout << "Enter the value of e:" << endl;
    cin >> e;
    int em[n][e];
    for(int i= 0; i<n; i++){
        for(int j=0; j<e; j++){
            cin >> em[i][j];
        }
    }
```

P.T.O

```
for(int i=0; i<n; i++){
  for(int j=0; j<c; j++){
      cout<< em[i][j] << " ";
    }
   cout<<endl;
  }
```

VAL = em[0][0]

```
i=0;
j=0;

while( i!=(r-1) || j!=(c-1)){
  // cout<< "(i, j)", cout<<"("<<i<<","<<j<<")"
  if( i<(r-1) && j<(c-1)){
      if(em[i][j+1] <= em[i+1][j])
          { j++ }
      else{ i++}
```

```
else if (i == (n-1)){
    j++; }
else{
    i++; }
cout << "(" << i << "," << j << ")";

~~VAL = em[i]~~
    VAL += em[i][j];
    cout << endl;
}

cout << VAL << endl;

return 0;
}
```

The pseudo code indicates minimum cost path problem.

The pseudo code in in Greedy form.

Ans: _____

2017000000040

Answer to the Question No. 02

C++

```
#include <bits/stdc++.h>

using namespace std;

int main(){
    int r, c;
    cin>>r>>c; // rown and colum input
    int arr2D[r][c];

    for(int i=0; i<c; i++){
        arr2D[r-1][i] = 1; }

    for( int i= 0; i<r; i++){
        arr2D[i][0] = 1;
    };
```

P.T.O

```
for(int i=r-2; i>=0; i--){
    for(int j=1; j<c; j++){
        if(i%2 != 0){   // ODD
            arr2D[i][j] = arr2D[r][j-1]
                        + arr2D[r+1][j];
        }
        else if(i%2==0){ // EVEN
            arr2D[i][j] = arr2D[r][j-1]
                        + arr2D[r+1][j]
                        + arr2D[r+1][j-1];
        }
    }
}
int countn = arr2D[0][c-1];
// Top rightmost cell of the 2D array
cout << " Number of ways to reach the
    top right cell from the bottom right
    cell is = " << countn << endl;
```

```
return 0;
}
```

Ans: