In [ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys



#
# Complete the 'calcMissing' function below.
#
# The function accepts STRING_ARRAY readings as parameter.
#
import datetime as dt
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
def calcMissing(readings):

   # to store data
    df1 = []
    # to store predictions
    r1  = []

    for i in range(readings_count):
        df = readings[i].strip().split('\t')
        df = list(filter(None, df))
        df1.append([str(c) for c in df])

    df1 = pd.DataFrame(df1)
    #train test split

    test = df1[df1[1].str.contains('Missing')]
    test.columns = ['X_test', 'Y_test']

    train = df1[~df1[1].str.contains('Missing')]
    train.columns = ['X', 'Y']
    train['X'] = train['X'].apply(lambda x:dt.datetime.strptime(x,'%m/%d/%Y %
H:%M:%S'))
    train['X'].apply(lambda x: x.strftime('%m%d%Y'))
    train['X'] = pd.to_datetime(train['X'])
    train['X'].apply(lambda x:x.toordinal())

    test['X_test'] = test['X_test'].apply(lambda x:dt.datetime.strptime(x,'%m/
%d/%Y %H:%M:%S'))
    test['X_test'].apply(lambda x: x.strftime('%m%d%Y'))
    test['X_test'] = pd.to_datetime(test['X_test'])
    test['X_test'].apply(lambda x:x.toordinal())


    rf1 = RandomForestRegressor(n_estimators=100, oob_score=True, random_state
=0)
```

```
    rf1.fit(train.loc[:,train.columns != 'Y'],train.loc[:,'Y'])
    for i,j in enumerate(rf1.predict(test['X_test'].values.reshape(-1,1))):
        r1.append((test['X_test'].index[i],round(j,1)))

    r1.sort()
    for k in range(len(r1)):
        print(r1[k][1])



if __name__ == '__main__':
```