Reading ECG data

In [1]:

```
import h5py
```

In [2]:

```
N = 10
print(f'The {N}-th record is records/A{N:05d}.h5')
```

```
The 10-th record is records/A00010.h5
```

In [3]:

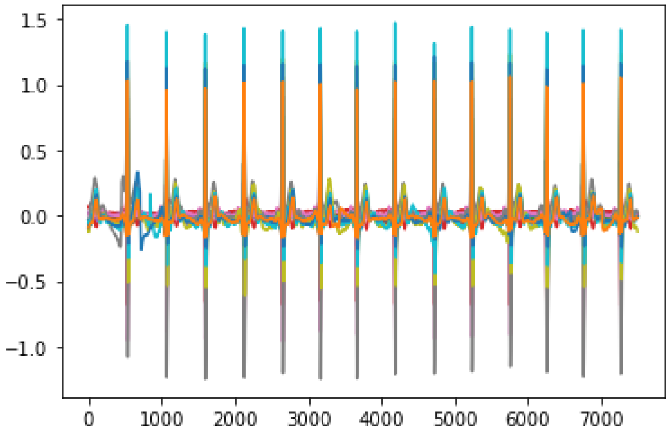
```
# read an ECG signal
with h5py.File(f'records/A{N:05d}.h5', 'r') as f:
    signal = f['ecg'][()]
    print('The size of record: {}'.format(signal.shape))
```

```
The size of record: (12, 7500)
```

In [4]:

```
import matplotlib.pyplot as plt

# plot the signal
for i in range(12):
    plt.plot(signal[i])
```



Reading metadata

In [5]:

```
import pandas as pd
```

In [6]:

```
# access the attributes
df = pd.read_csv('metadata.csv');
df[df.ECG_ID == f'A{N:05d}']
```

Out[6]:

	ECG_ID	AHA_Code	Patient_ID	Age	Sex	N	Date
9	A00010	1	S00010	32	F	7500	2020-06-29

In [7]:

```
def remove_nonprimary_code(x):
    """Remove non-primary statement"""
    r = []
    for cx in x:
        for c in cx.split('+'):
            if int(c) < 200 or int(c) >= 500:
                if c not in r:
                    r.append(c)
    return r
```

```
# obtain primary statements
codes = df.AHA_Code.str.split(';')
primary_codes = codes.apply(remove_nonprimary_code)
```

In [8]:

```
# get the diagnosis
desc = pd.read_csv('code.csv')
print('The diagnosis:')
for c in primary_codes[N-1]:
    print(desc[desc.Code == int(c)].Description.iloc[0])
```

```
The diagnosis:
Normal ECG
```

Signal quality assessment

In [9]:

```
from numpy.fft import fft, fftshift, fftfreq
import numpy as np
```

In [10]:

```
def SQI(ecg_lead):
    """Return basSQI and pSQI of an ECG lead."""
    L = ecg_lead.size
    fs = 500
    freq = fftshift(fftfreq(L, 1/fs))
    amp = fftshift(np.abs(fft(ecg_lead)/L))
    ind = round(amp.size/2 - 0.5)
    amp[ind+1:] *= 2

    freq = freq[ind:]
    amp = amp[ind:]

    # basSQI
    s = np.sum(amp[freq<=40]**2)
    s2 = np.sum(amp[freq<=1]**2)
    bas = 1-s2/s

    # pSQI
    s = np.sum(amp[(freq<=40)&(freq>=5)]**2)
    s2 = np.sum(amp[(freq<=15)&(freq>=5)]**2)
    p = s2/s
    return bas, p


def average_quality(signal):
    """Return the average basSQI and pSQI of a 12-lead ECG signal."""
    bas = p = 0
    for i in range(12):
        r1, r2 = SQI(signal[i])
        bas += r1
        p += r2
    return bas/12, p/12


def remove_nonprimary_code(x):
    """Remove non-primary statement"""
    r = []
    for cx in x:
        for c in cx.split('+'):
            if int(c) < 200 or int(c) >= 500:
                if c not in r:
                    r.append(c)
    return r
```

In [11]:

```
bas, p = average_quality(signal)
print('The quality of signal: basSQI = {:.3f}, pSQI = {:.3f}'.format(bas, p))
```

```
The quality of signal: basSQI = 0.950, pSQI = 0.707
```

Dataset splitting

In [12]:

```
# 80%-20% split
def ecg_train_test_split(df):
    # put all records belonging to patients with
    # multiple records in the test set
    test1 = df.Patient_ID.duplicated(keep=False)
    N = int(len(df)*0.2) - sum(test1)
    # 73 is chosen such that all primary statements exist in both sets
    df_test = pd.concat([df[test1], df[~test1].sample(N, random_state=73)])
    df_train = df.iloc[df.index.difference(df_test.index)]
    return df_train, df_test

df_train, df_test = ecg_train_test_split(df)
print(f'The training set has {len(df_train)} records')
print(f'The test set has {len(df_test)} records')
```

```
The training set has 20616 records
The test set has 5154 records
```

In [13]:

```
def flatten(lst):
    return [item for sublist in lst for item in sublist]

train_counts = pd.DataFrame(
    zip(*np.unique(
        flatten(df_train.AHA_Code.str.split(';').apply(
            remove_nonprimary_code).to_list()),
        return_counts=True)), columns=['Code', 'Count'])

test_counts = pd.DataFrame(
    zip(*np.unique(
        flatten(df_test.AHA_Code.str.split(';').apply(
            remove_nonprimary_code).to_list()),
        return_counts=True)), columns=['Code', 'Count'])

train_counts.set_index('Code').join(
    test_counts.set_index('Code'), lsuffix='_train', rsuffix='_test')
```

Out[13]:

	Count_train	Count_test
Code		
1	11295	2610
101	123	31
102	4	2
104	66	18
105	1012	247
106	515	195
108	20	7
120	127	34
121	107	31
125	252	70
140	15	4
142	158	51
143	5	1
145	1435	394
146	832	231
147	1736	482

	Count_train	Count_test
Code		
148	18	6
152	7	2
153	63	25
155	27	5
160	35	17
161	95	25
165	61	30
166	4	3
21	570	155
22	2161	550
23	1254	299
30	423	116
31	3	1
36	51	13
37	14	6
50	502	173
51	62	37
54	12	1
60	795	272
80	10	1
81	1	2
82	195	43
83	7	2
84	2	1
85	14	21
86	39	8
87	2	1
88	14	8