TASK 1

STEP 1

A VCF (variant call format file) file is the output of a bioinformatics pipeline. DNA sample is sequenced through NGS system and aligned to have SAM/BAM file. This SAM/BAM file is compared with a reference genome and stored the output in a VCF file.

A VCF file usually has three main sections:

i. Meta Information Lines – It describes the format and content of the file. Prefixed by double pound symbol (##).

ii. Header Line – A single line represents columns of data. Prefixed with a one pound symbol (#).

iii. Data Lines – It contains data corresponds to the columns specified in the header.

Meta information may contain the following:

- ##fileformat
- ##FILTER
- ##FORMAT
- ##INFO
- ##reference
- ##source

In the example_header.vcf:

Used analysis/pipeline is: CalculateGenotypePosteriors
Version of the human genome is: file:///humgen/1kg/reference/human_g1k_v37_decoy.fasta

## CODE

```
import re

with open("example_header.vcf", "r") as f:
    informations = f.read()

analysis_tool = re.findall("(?<=##source=)(.*)", informations)
ref_genome = re.findall("(?<=##reference=)(.*)", informations)

print(f"Used analysis/pipeline is: {analysis_tool[0]}")
print(f"Version of the human genome is: {ref_genome[0]}")
```

STEP 2

For the file ‘trio_example.vcf’ :

- ✔ Number of SNVs are: 558686
- ✔ Number of SNSs are: 497631
- ✔ Number of InDels are: 61055

The results are included for each individual chromosome in the following table:

Chromosome	SNSs	INDELs	SNVs
7	351468	42818	394286
20	134352	16250	150602
X	11811	1987	13798

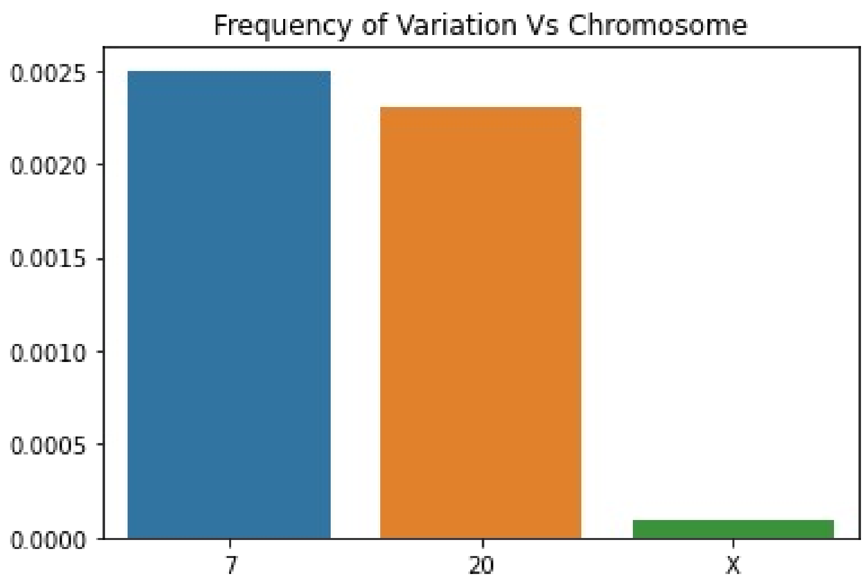
## CODE

```
import re
from collections import defaultdict

with open("trio_example.vcf", "r") as f:
    data = re.findall("(?=#CHROM).*", f.read(), re.M|re.S)

records = defaultdict(dict)
indels, sns = 0, 0
for line in data[0].strip().splitlines()[1:]:
    line = list(map(str.strip, line.split("\t")))
    ref_len, alt_len = len(line[3]), len(line[4])
    if ref_len == alt_len == 1:
        if 'SNS' not in records[line[0]]:
            records[line[0]]['SNS'] = 0
        records[line[0]]['SNS'] += 1
    elif ref_len!=alt_len:
        if 'INDEL' not in records[line[0]]:
            records[line[0]]['INDEL'] = 0
        records[line[0]]['INDEL'] += 1

for k, v in records.items():
    records[k]['SNV'] = v['SNS'] + v['INDEL']

print(records)
```

STEP 3

Frequency of SNV for each chromosome are listed in the following table for “trio_example.vcf”:

Chromosome	Length	Frequency of variation
7	159345973	0.0025
20	64444167	0.0023
X	156040895	0.0001



We don’t have the following chromosomes in the file:
1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, Y

The variance are not uniformly distributed.

## CODE

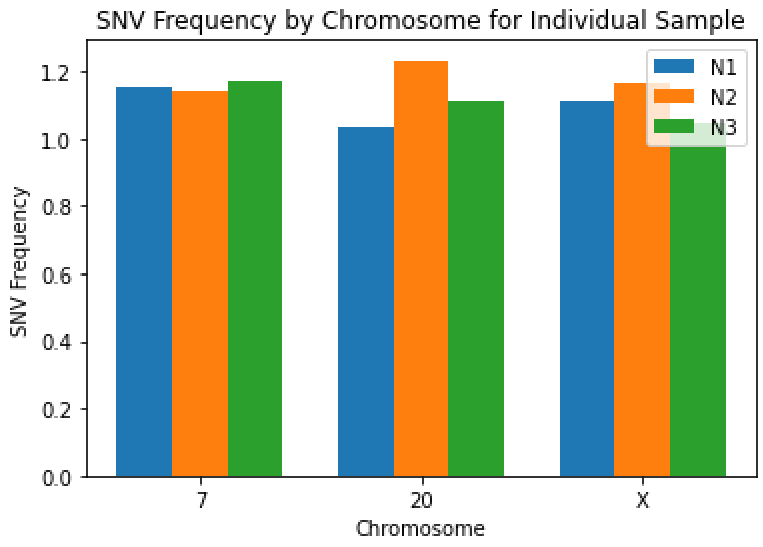
```
with open("trio_example.vcf", "r") as f:
    data = re.findall("(?=#CHROM).*", f.read(), re.M|re.S)

with open("chromosome_lengths.tsv", "r") as f:
    chr_len = {}
    for line in f.readlines()[1:]:
        line = line.strip().split("\t")
        chr_len[line[0]] = int(line[1])

frequency = {}
snvs = 0
for line in data[0].strip().splitlines()[1:]:
    line = list(map(str.strip, line.split("\t")))
    ref_len, alt_len = len(line[3]), len(line[4])
    if (ref_len == alt_len == 1) or (ref_len!=alt_len):
        frequency[line[0]] = frequency.get(line[0], 0) + 1
        snvs += 1

frequency_of_variation = {k: round(v/chr_len[k], 4) for k, v in frequency.items()}
print(frequency_of_variation)
```

TASK 2

STEP 4

Homozygous to Heterozygous ratio for each chromosome in individual sample are listed in the following table:

Sample/Chromosome	7	20	X
N1	1.151	1.035	1.111
N2	1.141	1.233	1.168
N3	1.169	1.111	1.045



It is not possible to determine the sex of the samples based on the above data alone.

The ratio of homozygous to heterozygous genotypes for each sample and chromosome provides information about the genetic diversity within each sample, but it does not provide information about the sex of the individual. In humans, sex is determined by the presence of sex chromosomes. Females have two X chromosomes (XX), while males have one X and one Y chromosome (XY). However, the given data only includes three chromosomes (7, 20, and X) and does not provide information about the presence or absence of Y chromosomes, which is necessary to determine the sex of an individual.

##CODE

```
import re
from collections import defaultdict

with open("trio_example.vcf", "r") as f:
    data = re.findall("(?=#CHROM).*", f.read(), re.M|re.S)

samples = data[0].splitlines()[0].strip().split("\t")[9:]
# Define the nested defaultdict
new_dict = lambda: defaultdict(lambda: defaultdict(dict))
records = new_dict()
count = 0
for iline in data[0].strip().splitlines()[1:]:
    line = list(map(str.strip, iline.split("\t")))
    count += 1
    for sample, h in zip(samples, line[9:]):
        h1, h2 = list(map(int, h.split("/")))
        if h1 == h2:
            if "homo" not in records[sample][line[0]].keys():
                records[sample][line[0]]['homo'] = 0
            records[sample][line[0]]['homo'] += 1
        else:
            if "hetero" not in records[sample][line[0]].keys():
                records[sample][line[0]]['hetero'] = 0
            records[sample][line[0]]['hetero'] += 1

ratios = defaultdict(dict)
for k1, v1 in records.items():
    for k2, v2 in v1.items():
        ratios[k1][k2] = round(v2['homo'] / v2['hetero'], 3)
print(ratios)
```

STEP 5

Count of De Novo mutations for each individual samples are listed in the following table:

	SNP	INDEL
N1	34401	4489
N2	199018	22939
N3	37016	4726

No, it is not possible to identify a child from this data alone.
The given data appears to be related to genetic variations (SNP and INDEL) found in multiple individuals (N1, N2, N3). Without additional information such as the genetic makeup or relationship between these individuals, it is not possible to identify a child from this data. One common approach is to use software tools such as PLINK or KING to perform kinship analysis on the VCF data.

##CODE

```
with open("trio_example.vcf", "r") as f:
    data = re.findall("(?=#CHROM).*", f.read(), re.M|re.S)

samples = data[0].splitlines()[0].strip().split("\t")[9:]

def extract_gt(gt):
    return int(gt.split("/")[0])

denovos = defaultdict(dict)
for iline in data[0].strip().splitlines()[1:]:
    line = list(map(str.strip, iline.split("\t")))
    for i in range(3):
        sample_id = samples[i]
        if i == 0:
            child_gt, father_gt, mother_gt = list(map(extract_gt, [line[9],
line[10], line[11]]))
        elif i == 1:
            child_gt, father_gt, mother_gt = list(map(extract_gt, [line[10],
line[11], line[9]]))
        else:
            child_gt, father_gt, mother_gt = list(map(extract_gt, [line[11],
line[9], line[10]]))

        # Check if the variant is a candidate de novo mutation
        if child_gt != father_gt and child_gt != mother_gt:
            # Increment the count for the individual and mutation type
            mutation_type = 'SNP' if len(line[3]) == len(line[4]) == 1 else 'INDEL'

            if mutation_type not in denovos[sample_id].keys():
                denovos[sample_id][mutation_type] = 0
            denovos[sample_id][mutation_type] += 1

print(denovos)
```