Kulvir Singh

19BCE2074

Digital Assignment 2

Question 1

Problem Statement :

Use beautiful soup to crawl any one of the E-commerce websites of your choice and perform the same. The following information needs to be extracted from the page: (Choose any one product: e.g. laptop, smartphone … etc.)

a)Product name

b)Product price

c)Product discount

d)Product image

Procedure :

Tools Used: Beautiful Soup(bs4), request packages, python, ide to run the program (Google Colab) for execution of various commands
Steps for Implementation:
1. Install the bs4, request packages (pip install bs4, request)
2. Store the URL to be scraped as a string
3. use request.get to get data from URL
4. use html parser for extracting the data as html
5. use find and accurate tags to extract the required data
6. run the code in the colab terminal

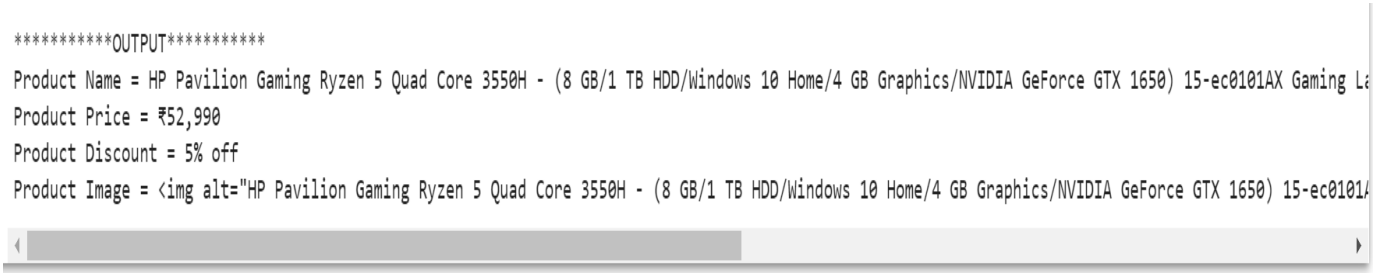
Code :

```
!pip install requests
!pip install bs4

import requests
from bs4 import BeautifulSoup

webpageurl = "https://www.flipkart.com/hp-pavilion-gaming-ryzen-5-quad-core-3550h-8-gb-1-tb-hdd-windows-10-home-4-gb-graphics-nvidia-geforce-gtx-1650-15-ec0101ax-laptop/p/itma1af6bf593dc8?pid=COMFSFNVDXG74QXR&lid=LSTCOMFSFNVDXG74QXRY8FRH2&marketplace=FLIPKART&srno=b_1_22&otracker=hp_rich_navigation_6_1.navigationCard.RICH_NAVIGATION_Electronics~Gaming_PK7I48M403OR&fm=organic&iid=a4a75fb6-a7c4-406f-8829-6f0897fc37fc.COMFSFNVDXG74QXR.SEARCH&ppt=browse&ppn=browse&ssid=tri79mziu80000001614107514263"
data = requests.get(webpageurl)

soup = BeautifulSoup(data.text, 'html.parser')

product_name = soup.find('span', {'class': 'B_NuCI'})
product_price = soup.find('div', {'class': '_30jeq3 _16Jk6d'})
product_discount = soup.find('div', {'class': '_3Ay6Sb _31Dcoz'})
product_image = soup.find('div', {'class': 'CXW8mj _3nMexc'})

print("\n\n**********OUTPUT**********")
print("Product Name = "+product_name.text)
print("Product Price = "+product_price.text)
print("Product Discount = "+product_discount.span.text)
print("Product Image = "+str(product_image.img))
```

Code Screenshot:

```
!pip install requests
!pip install bs4

import requests
from bs4 import BeautifulSoup

webpageurl = "https://www.flipkart.com/hp-pavilion-gaming-ryzen-5-quad-core-3550h-8-g
data = requests.get(webpageurl)

soup = BeautifulSoup(data.text, 'html.parser')

product_name = soup.find('span', {'class': 'B_NuCI'})
product_price = soup.find('div', {'class': '_30jeq3 _16Jk6d'})
product_discount = soup.find('div', {'class': '_3Ay6Sb _31Dcoz'})
product_image = soup.find('div', {'class': 'CXW8mj _3nMexc'})

print("\n\n**********OUTPUT**********")
print("Product Name = "+product_name.text)
print("Product Price = "+product_price.text)
print("Product Discount = "+product_discount.span.text)
print("Product Image = "+str(product_image.img))
```

Output Screenshots :



Output as text

```
**********OUTPUT**********
Product Name = HP Pavilion Gaming Ryzen 5 Quad Core 3550H - (8 GB/1 TB
HDD/Windows 10 Home/4 GB Graphics/NVIDIA GeForce GTX 1650) 15-ec0101AX
Gaming Laptop  (15.6 inch, Black, 2.04 kg)
Product Price = ₹52,990
Product Discount = 5% off
Product Image = img alt="HP Pavilion Gaming Ryzen 5 Quad Core 3550H - (8
GB/1 TB HDD/Windows 10 Home/4 GB Graphics/NVIDIA GeForce GTX 1650) 15-
ec0101AX Gaming Laptop" class="_396cs4 _2amPTt _3qGmMb"
src="//img1a.flixcart.com/www/linchpin/fk-cp-
zion/img/placeholder_fcebae.svg"/
```

Question 2

Problem Statement :

Write a python program to find the important words from the text using TF-IDF. Use minimum of 5 documents with the real text source from a web page of some relevance.

Procedure :

1. We need to count the imp terms (i.e. words that occur the most in the doc)
2. We will use TextBlob for breaking up text to words and get a word count. We will count the freq of words in a document, normalized by dividing with the number of words in the doc.
3. n_containing will return the number of occurrences of word in different documents.
4. idf(word, bloblist) computes "inverse document frequency" which measures how common a word is among all documents in bloblist. The more common a word is, the lower its idf. We take the ratio of the total number of documents to the number of documents containing word, then take the log of that. Add 1 to the divisor to prevent division by zero.
5. tfidf(word, blob, bloblist) computes the TF-IDF score. It's the product of tf and idf.
6. For each document, we store the TF-IDF scores in a dictionary scores mapping word => score using a dict comprehension. We then sort the words by their scores and output the top 3 words.

Code :

```
import nltk
nltk.download('punkt')
import math
from textblob import TextBlob as tb
print('\n***OUTPUT***\n')
def tf(word, blob):
 return blob.words.count(word) / len(blob.words)
def n_containing(word, bloblist):
 return sum(1 for blob in bloblist if word in blob)
def idf(word, bloblist):
 return math.log(len(bloblist) / (1 + n_containing(word, bloblist)))
def tfidf(word, blob, bloblist):
 return tf(word, blob) * idf(word, bloblist)
document1 = tb("""Python is a 2000 made-for-
TV horror movie directed by Richard Clabaugh. The film features several cu
lt favorite actors, including William Zabka of The Karate Kid fame, Wil Wh
eaton, Casper Van Dien, Jenny McCarthy, Keith Coogan, Robert Englund (best
 known for his role as Freddy Krueger in the A Nightmare on Elm Street ser
ies of films), Dana Barron, David Bowe, and Sean Whalen. The film concerns
 a genetically engineered snake, a python, that escapes and unleashes itse
lf on a small town. It includes the classic final girl scenario evident in
```

```
 films like Friday the 13th. It was filmed in Los Angeles, California and
Malibu, California. Python was followed by two sequels: Python II (2002) a
nd Boa vs. Python (2004), both also made-for-TV films.""")
document2 = tb("""Python, from the Greek word (πύθων/πύθωναϛ), is a genus
of nonvenomous pythons[2] found in Africa and Asia. Currently, 7 species a
re recognised.[2] A member of this genus, P. reticulatus, is among the lon
gest snakes known.""")
document3 = tb("""The Colt Python is a .357 Magnum caliber revolver former
ly manufactured by Colt's Manufacturing Company of Hartford, Connecticut.
It is sometimes referred to as a "Combat Magnum".[1] It was first introduc
ed in 1955, the same year as Smith & Wesson's M29 .44 Magnum. The now disc
ontinued Colt Python targeted the premium revolver market segment. Some fi
rearm collectors and writers such as Jeff Cooper, Ian V. Hogg, Chuck Hawks
, Leroy Thompson, Renee Smeets and Martin Dougherty have described the Pyt
hon as the finest production revolver ever made.""")
document4 = tb("""Web mining is the application of data mining techniques
to discover patterns from the World Wide Web. As the name proposes, this i
s information gathered by mining the web. It makes utilization of automate
d apparatuses to reveal and extricate data from servers and web2 reports,
and it permits organizations to get to both and unstructured information f
rom browser activities, server logs, website and link structure, page cont
ent and different sources.""")
document5 =tb("""In computing, a hyperlink, or simply a link, is a referen
ce to data that the user can follow by clicking or tapping.[1] A hyperlink
 points to a whole document or to a specific element within a document. Hy
pertext is text with hyperlinks. The text that is linked from is called an
chor text. A software system that is used for viewing and creating hyperte
xt is a hypertext system, and to create a hyperlink is to hyperlink (or si
mply to link). A user following hyperlinks is said to navigate or browse t
he hypertext.""")
bloblist = [document1, document2, document3, document4, document5]
for i, blob in enumerate(bloblist):
 print("Top words in document {}".format(i + 1))
 scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
 sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
 for word, score in sorted_words[:3]:
  print("Word: {}, TF-IDF: {}".format(word, round(score, 5)))
```

Code Screenshot:

```
import nltk
nltk.download('punkt')
import math
from textblob import TextBlob as tb
print('\n***OUTPUT***\n')
def tf(word, blob):
 return blob.words.count(word) / len(blob.words)
def n_containing(word, bloblist):
 return sum(1 for blob in bloblist if word in blob)
def idf(word, bloblist):
 return math.log(len(bloblist) / (1 + n_containing(word, bloblist)))
def tfidf(word, blob, bloblist):
 return tf(word, blob) * idf(word, bloblist)
document1 = tb("""Python is a 2000 made-for-TV horror movie directed by Richard Clabaugh. The film features several cult favorite actors, including W
document2 = tb("""Python, from the Greek word (πύθων/πύθωνας), is a genus of nonvenomous pythons[2] found in Africa and Asia. Currently, 7 species ar
document3 = tb("""The Colt Python is a .357 Magnum caliber revolver formerly manufactured by Colt's Manufacturing Company of Hartford, Connecticut. I
document4 = tb("""Web mining is the application of data mining techniques to discover patterns from the World Wide Web. As the name proposes, this is
document5 =tb("""In computing, a hyperlink, or simply a link, is a reference to data that the user can follow by clicking or tapping.[1] A hyperlink
bloblist = [document1, document2, document3, document4, document5]
for i, blob in enumerate(bloblist):
 print("Top words in document {}".format(i + 1))
 scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
 sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
 for word, score in sorted_words[:3]:
  print("Word: {}, TF-IDF: {}".format(word, round(score, 5)))
```

Output Screenshots :

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

***OUTPUT***

Top words in document 1
Word: films, TF-IDF: 0.02253
Word: python, TF-IDF: 0.02094
Word: made-for-TV, TF-IDF: 0.01502
Top words in document 2
Word: genus, TF-IDF: 0.04953
Word: Greek, TF-IDF: 0.02476
Word: word, TF-IDF: 0.02476
Top words in document 3
Word: Colt, TF-IDF: 0.03124
Word: Magnum, TF-IDF: 0.03124
Word: revolver, TF-IDF: 0.03124
Top words in document 4
Word: Web, TF-IDF: 0.03927
Word: mining, TF-IDF: 0.03927
Word: web, TF-IDF: 0.03927
Top words in document 5
Word: hyperlink, TF-IDF: 0.03984
Word: Hypertext, TF-IDF: 0.03984
Word: hypertext, TF-IDF: 0.03984
```

Question 3

Problem Statement :

Write a python program to perform the following encoding and decoding for the even numbers from 1 – 20.

i) Elias Gamma
ii) Elias Delta
iii) Golomb (b = 10)

Procedure :

Elias Gamma Encoding

Write *x* in binary. Subtract 1 from the number of bits written in step 1 and prepend that many zeros. Use basic python libraries and functions to perform the same.

Elias Gamma Decoding

Read and count zeroes from the stream until we reach the first one. Call this count of zeroes *K*. Consider the one that was reached to be the first digit of the integer, with a value of 2*K*, read the remaining *K* bits of the integer. Use basic python libraries and functions to perform the same.

Elias Delta Encoding

Let N = ⌊log2 X⌋; be the highest power of 2 in X, so 2N ≤ X < 2N+1. Let L = ⌊log2 N+1⌋ be the highest power of 2 in N+1, so 2L ≤ N+1 < 2L+1. Write L zeros, followed by the L+1-bit binary representation of N+1, followed by all but the leading bit (i.e. the last N bits) of X.

Elias Delta Decoding

Read and count zeros from the stream until you reach the first one. Call this count of zeros L. Considering the one that was reached to be the first digit of an integer, with a value of 2L, read the remaining L digits of the integer. Call this integer N+1, and subtract one to get N. Put a one in the first place of our final output, representing the value 2N. Read and append the following N digits.

Golomb Encoding

To Golomb-code a number, find the quotient and remainder of division by the divisor. Write the quotient in unary notation, then the remainder in truncated binary notation. In practice, you need a stop bit after the quotient: if the quotient is written as a sequence of zeroes, the stop bit is a one (or vice versa - and people do seem to prefer to write their unary numbers with ones, which is Wrong). The length of the remainder can be determined from the divisor.

Golomb Decoding

k ←⌈log2(m)⌉. t ← 2k − m. Let s ← the number of consecutive ones in the input (we stop when we read a 0). Let x ← the next k − 1 bits in the input. If x < t: s ← s × m + x. Else: x ← x × 2+ next input bit. s ← s × m + x − t.

Code :

Elias Gamma

```
from math import log
log2 = lambda x: log(x, 2)

def Unary(x):
    return (x-1)*'0'+'1'

def Binary(x, l = 1):
    s = '{0:0%db}' % l
    return s.format(x)

def binaryToDecimal(binary):
    binary1 = binary
    decimal, i, n = 0, 0, 0
    while(binary != 0):
        dec = binary % 10
        decimal = decimal + dec * pow(2, i)
        binary = binary//10
        i += 1
    return decimal

def Elias_Gamma(x):
    if(x == 0):
        return '0'

    n = 1 + int(log2(x))
    b = x - 2**(int(log2(x)))

    l = int(log2(x))

    return Unary(n) + Binary(b, l)
ec=[]
even_num = [2,4,6,8,10,12,14,16,18,20]
for i in even_num:
  ec.append(Elias_Gamma(i))
  print('Elias Gamma of '+str(i)+' : '+(Elias_Gamma(i)))
```

```
for i in ec:
  binstr = (i[i.index('1'):])
  binstr = int(binstr)
  print('Original Value of '+str(i)+' : '+str(binaryToDecimal(binstr)))
```

Elias Delta

```
import math

def gamma(t):
    x=[];
    y=[];
    while(t>0):
        x.append(t%2);
        t=int(t/2);
    for i in range(len(x)-1):
        y.append(0);
    for i in range(len(x)):
        y.append(x.pop());
    return y;

even_num = [2,4,6,8,10,12,14,16,18,20]
ed = []
for x in even_num:
 temp = str(x)
 t=math.floor(1+math.log(x,2));
 p=gamma(t);
 y=[];
 while(x>0):
     y.append(x%2);
     x=int(x/2);
 y.pop();
 for i in range(len(y)):
     p.append(y.pop());
 a=''.join(map(str,p));
 ed.append(a)
 print('Elias Delta Code of '+temp+' : '+a)

 def decode(x):
    num=0;
    for i in range(len(x)):
        num+=(int(x[len(x)-1-i])*(math.pow(2,i)));
    return num;
```

```
for x in ed:
 if(x=='1'):
     print('1');
     exit;
 else:
     x=list(x);
     t=0;
     v=[];
     b=0;
     w=[];
     c=0;
     for i in x:
         if(b!=1):
             if(i=='0'):
                 t+=1;
             else:
                 v.append(i);
                 b=1;
         elif(c!=1):
             if(t==0):
                 c=1;
                 w.append('1');
                 w.append(i);
             else:
                 v.append(i);
                 t-=1;
         else:
             num=decode(v);
             if(num==0):
                 break;
             else:
                 w.append(i);
                 num-=1;
     ans=decode(w);
     print('Original Number: ',int(ans));
```

Golomb Code:

```
import math
def listToString(l) :
  s=""
  for i in l:
    s = s+str(i)
  return s
def encode(t):
```

```
    x=[];
    if(t==0):
        return [0];
    while(t>0):
        x.append(t%2);
        t=int(t/2);
    return x
def unary(t):
    y=[];
    for i in range(t-1):
        y.append(0);
    y.append(1)
    return y;
GolombCode = []
even_num=[2,4,6,8,10,12,14,16,18,20]
b=10
for x in even_num:
 q=int(x/b)
 y=unary(q+1)
 r=x-(q*b)
 i=math.floor(math.log(b,2));
 d=math.pow(2,i+1)-b;
 if(r>=d):
     r+=int(d);
 r2=encode(r);
 if(len(r2)<=i and r>=d):
     r2.append(0);
 if(len(r2)<i and r<d):
     r2.append(0);
 r2=r2[::-1];
 y=y+r2;
 GolombCode.append(listToString(y));
 print('Code of '+str(x)+': '+listToString(y))
def decode(x):
    num=0;
    for i in range(len(x)):
        num+=(int(x[len(x)-1-i])*(math.pow(2,i)));
    return num;
for i in range(10):
 x=str(input('Enter code: '))
 x=list(x)
 i=math.floor(math.log(b,2))
 d=math.pow(2,i+1)-b
 p2=0;
 l=1;
```

```
while(p2<len(x)):
    t=0;
    flag=0;
    r=[];
    k=i;
    q=0;
    for p in range(p2,len(x)):
        if(x[p]=='0' and flag==0):
            t+=1;
            continue;
        if(x[p]=='1' and flag==0):
            q=t;
            flag=1;
            continue;
        r.append(x[p]);
        k-=1;
        if(k==0):
            rnum=decode(r);
            if(rnum<d):
                p2=p+1;
                break;
        if(k==-1):
            rnum=decode(r);
            rnum=rnum-d;
            p2=p+1;
            break;
    ans=q*b+rnum;
    print(ans);
    l=0;
```

Code Screenshot:

Elias Gamma

```
from math import log
log2 = lambda x: log(x, 2)
def Unary(x):
    return (x-1)*'0'+'1'
def Binary(x, l = 1):
    s = '{0:0%db}' % l
    return s.format(x)
def binaryToDecimal(binary):
    binary1 = binary
    decimal, i, n = 0, 0, 0
    while(binary != 0):
        dec = binary % 10
        decimal = decimal + dec * pow(2, i)
        binary = binary//10
        i += 1
    return decimal
def Elias_Gamma(x):
    if(x == 0):
        return '0'

    n = 1 + int(log2(x))
    b = x - 2**(int(log2(x)))

    l = int(log2(x))

    return Unary(n) + Binary(b, l)
ec=[]
even_num = [2,4,6,8,10,12,14,16,18,20]
for i in even_num:
  ec.append(Elias_Gamma(i))
  print('Elias Gamma of '+str(i)+' : '+(Elias_Gamma(i)))

for i in ec:
  binstr = (i[i.index('1'):])
  binstr = int(binstr)
  print('Original Value of '+str(i)+' : '+str(binaryToDecimal(binstr)))
```

Elias Delta

```
import math
def gamma(t):
    x=[];
    y=[];
    while(t>0):
        x.append(t%2);
        t=int(t/2);
    for i in range(len(x)-1):
        y.append(0);
    for i in range(len(x)):
        y.append(x.pop());
    return y;
even_num = [2,4,6,8,10,12,14,16,18,20]
ed = []
for x in even_num:
 temp = str(x)
 t=math.floor(1+math.log(x,2));
 p=gamma(t);
 y=[];
 while(x>0):
     y.append(x%2);
     x=int(x/2);
 y.pop();
 for i in range(len(y)):
     p.append(y.pop());
 a=''.join(map(str,p));
 ed.append(a)
 print('Elias Delta Code of '+temp+' : '+a)
 def decode(x):
    num=0;
    for i in range(len(x)):
        num+=(int(x[len(x)-1-i])*(math.pow(2,i)));
    return num;
for x in ed:
 if(x=='1'):
     print('1');
     exit;
 else:
     x=list(x);
     t=0;
     v=[];
     b=0;
     w=[];
     c=0;
     for i in x:
         if(b!=1):
             if(i=='0'):
                 t+=1;
             else:
                 v.append(i);
                 b=1;
         elif(c!=1):
             if(t==0):
                 c=1;
                 w.append('1');
                 w.append(i);
             else:
                 v.append(i);
                 t-=1;
         else:
             num=decode(v);
             if(num==0):
                 break;
             else:
                 w.append(i);
                 num-=1;
     ans=decode(w);
     print('Original Number: ',int(ans));
```

Golomb Code:

```
import math
def listToString(l) :
  s=""
  for i in l:
    s = s+str(i)
  return s
def encode(t):
    x=[];
    if(t==0):
        return [0];
    while(t>0):
        x.append(t%2);
        t=int(t/2);
    return x
def unary(t):
    y=[];
    for i in range(t-1):
        y.append(0);
    y.append(1)
    return y;
GolombCode = []
even_num=[2,4,6,8,10,12,14,16,18,20]
b=10
for x in even_num:
 q=int(x/b)
 y=unary(q+1)
 r=x-(q*b)
 i=math.floor(math.log(b,2));
 d=math.pow(2,i+1)-b;
 if(r>=d):
     r+=int(d);
 r2=encode(r);
 if(len(r2)<=i and r>=d):
     r2.append(0);
 if(len(r2)<i and r<d):
     r2.append(0);
 r2=r2[::-1];
 y=y+r2;
 GolombCode.append(listToString(y));
 print('Code of '+str(x)+': '+listToString(y))
def decode(x):
    num=0;
    for i in range(len(x)):
        num+=(int(x[len(x)-1-i])*(math.pow(2,i)));
    return num;
for i in range(10):
 x=str(input('Enter code: '))
 x=list(x)
 i=math.floor(math.log(b,2))
 d=math.pow(2,i+1)-b
 p2=0;
 l=1;
 while(p2<len(x)):
     t=0;
     flag=0;
     r=[];
     k=i;
     q=0;
     for p in range(p2,len(x)):
         if(x[p]=='0' and flag==0):
             t+=1;
             continue;
         if(x[p]=='1' and flag==0):
             q=t;
             flag=1;
             continue;
         r.append(x[p]);
         k-=1;
         if(k==0):
             rnum=decode(r);
             if(rnum<d):
                 p2=p+1;
                 break;
         if(k==-1):
             rnum=decode(r);
             rnum=rnum-d;
             p2=p+1;
             break;
     ans=q*b+rnum;
     print(ans);
     l=0;
```

Output Screenshots :

Elias Gamma

```
Elias Gamma of 2 : 010
Elias Gamma of 4 : 00100
Elias Gamma of 6 : 00110
Elias Gamma of 8 : 0001000
Elias Gamma of 10 : 0001010
Elias Gamma of 12 : 0001100
Elias Gamma of 14 : 0001110
Elias Gamma of 16 : 000010000
Elias Gamma of 18 : 000010010
Elias Gamma of 20 : 000010100
Original Value of 010 : 2
Original Value of 00100 : 4
Original Value of 00110 : 6
Original Value of 0001000 : 8
Original Value of 0001010 : 10
Original Value of 0001100 : 12
Original Value of 0001110 : 14
Original Value of 000010000 : 16
Original Value of 000010010 : 18
Original Value of 000010100 : 20
```

Elias Delta

```
Elias Delta Code of 2 : 0100
Elias Delta Code of 4 : 01100
Elias Delta Code of 6 : 01110
Elias Delta Code of 8 : 00100000
Elias Delta Code of 10 : 00100010
Elias Delta Code of 12 : 00100100
Elias Delta Code of 14 : 00100110
Elias Delta Code of 16 : 001010000
Elias Delta Code of 18 : 001010010
Elias Delta Code of 20 : 001010100
Original Number:  2
Original Number:  4
Original Number:  6
Original Number:  8
Original Number:  10
Original Number:  12
Original Number:  14
Original Number:  16
Original Number:  18
Original Number:  20
```

Golomb Code:

```
Code of 2: 1010
Code of 4: 1100
Code of 6: 11100
Code of 8: 11110
Code of 10: 0100
Code of 12: 01010
Code of 14: 01100
Code of 16: 011100
Code of 18: 011110
Code of 20: 00100
Enter code: 1010
2.0
Enter code: 1100
4.0
Enter code: 11100
6.0
Enter code: 11110
8.0
Enter code: 01010
12.0
Enter code: 01100
14.0
Enter code: 011100
16.0
Enter code: 011110
18.0
```