ОГЛАВЛЕНИЕ

Семестр I

Тест 1. Списки (18 сентября)

Каков будет результат вызова ячейки с кодом
lst2 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
lst2[1] = [10, 11, 12, 13]
lst2
a. [[1 2, 3, 13], [10, 11, 12, 13], [7, 8, 9, 13]]
b. [[1 10, 3], [4, 11, 6], [7, 12, 9], [13]]
c. Сообщение об ошибке
d. [[1 , 2, 3], [10, 11 12], [7, 8, 9]]
e. [1, 2, 3, None], [10, 11, 12, 13], [7, 8, 9, None]]
f. [[1, 2, 3], [10, 11, 12, 13], [7, 8, 9]]

Каков будет результат вызова ячейки с кодом:
lst3 = []
for s in '1 2':
    lst3.append(s)
else:
    lst3.append('a')
lst3
a. ['1 2', 'a']
b. ['1', '2', 'a']
c. ['1', '', '2']
d. ['1', ' ', '2', 'a']
e. ['1', '2']

Тест 2. Словари. Множества. Кортежи (2 октября)

#Что выведет на экран ячейка с кодом:
d2 = dict ([1,2] , [3,4]])
print(d2.get(2,5))
Выберите один ответ:
a. 2
b. Сообщение об ошибке
c. 1
d. 5
e. (3,4)
f.4

#Что выведет на экран ячейка с кодом:
d2 = dict (zip(range(3), 'abc'))
print(d1.get('c', 3))
Выберите один ответ:
a. Сообщение об ошибке
b.(2, 'c')
c.'c'
d.2
e.3
f.None

#Что выведет на экран ячейка с кодом:
d1 = dict(zip('abc', range(2)))
print(d1.get('c'))
Выберите один ответ:
a.(2, 'c')
b.сообщение об ошибке
c.'c'
d.'a'
e.2
f.None

s1=set([1,3,5,3]) - set(range(4))
s1.pop()
print(len(s1))
Выберите один ответ:
a.{5}
b.set()
c.2
d.1
e.0

#Что выведет на экран ячейка с кодом:
fl = [1,3,0,4,1]
for i, e in enumerate (fl):
if e%2 ==1:
del fl [i]
fl
Выберите один ответ:
a.[3,0,4]
b.[1,0,1]
c.[1,3,1]
d.Сообщение об ошибке
e.[0,4]
f.[1,3,0,4,1]

Тест 3. Функции (16 октября)

def m_sum (x=’_’, y=’-’, z=’^’):
return x +2*y + 3*z
#что вернет вызов функции:
m_sum(*’ab’)
Выберите один ответ:
a.сообщение об ошибке
b._ab^^^
c.abb3^
d.abb
e.abb^^^
f._-ababab

def m_sum (x=1, y=1, z=1):
return x+2*y+3*z
#что вернет вызов функции:
m_sum(*’ab’)
Выберите один ответ:
a.’abc23’
b.’12abc’
c.’abbccc’
d.’a2y3z’
e.сообщение об ошибке
f.’123’

def f1(a,b):
return[a]+b
a = f1
f1(‘a’, a (‘b’, []))
Выберите один ответ:
a.[‘a’,’b’]
b.[[‘a’], [‘b’], []]
c.[‘a’, f1, ‘b’, []]
d.сообщение об ошибке
e.[‘a’,’b’, None]
f.[‘a’, a, ‘b’]

set (abs(x-3) for x in range(7))
Выберите один ответ:
a.{0,1,2,3}
b.[3,2,1,0,1,2,3]

c.{0,1,2,3,4}
d.{-3,-2,-1,0,1,2,3}
e.сообщение об ошибке
f.{3,2,1,0,1,2,3}

```
def func (ab):
        ab = 5
        global gl
        gl = 15
        ab = 10
gl = 10
ab = 20
func(gl)
ab = 0
print(gl) #что будет выведено на экран?
```

Выберите один ответ:
a.0
b.20
c.10
d.5
e.сообщение об ошибке
f.15

```
def super_iterator(a,b):
        ai = iter(a)
        bi = iter(b)
        try:
                while True:
                        yield next(ai)
                        yield next(bi)
        except StopIteration:
                pass
```

Каков результат вызова: list(super_iterator([1,2,3,4],[10,11]))
a.[1,2,3,4,10,11]
b.сообщение об ошибке
c.[1,10,2,11,3]
d.[1,10]
e.[1,10,2,11,3,11,4]
f.[1,10,2,11,3,4]

```
#Что выведет на экран ячейка с кодом:
sl = ['abc', 'de', 'f']
```

[b for a in sl for b in a]
Выберите один ответ:
a.[‘abc’, ‘de’, ‘f’]
b.сообщение об ошибке
c.[‘abcdef’]
d.[‘a’,’b’,’c’,’d’,’e’,’f’]
e.[[‘abc’],[‘de’],[‘f’]]
f.’abcdef’

```
def m_sum (x=1, y=1, z=1):
        return x+2*y+3*z
#Что вернет вызов функции:
m_sum (*’abcd’)
```

Выберите один ответ:
a.’abbcdcdcd’
b.’12abcd’
c.’abbccc’
d.’abccddd’
e.сообщение об ошибке
f.’abcd23’

```
def f2(f,s):
        return [f(el) for el in s]
f2(lambda v: v*2, ‘abc’)
```

Выберите один ответ:
a.сообщение об ошибке
b.[‘a’,’a’,’b’,’b’,’c’,’c’]
c.’aabbcc’
d.’abcabc’
e.[‘aa’, ‘bb’, ‘cc’]
f.[‘a’,’b’,’c’,’a’,’b’,’c’]

```
def f(a)
        def g(b):
                c = a+b
                print(c)
                return c
        return g
#Чем является функция f?
```

Выберите один ответ:
a.Функцией без побочных эффектов
b.Чистой функцией

c.Рекурсивной функцией
d.Лямбда-функцией
e.Замыканием
f.Функцией, написанной с ошибкой

Тест 4. Файлы и исключения (30 октября)

Что будет выведено на экран?
txt = ‘abc\ndef’

with open (‘’my2.txt’’,’’wt’’) as f:
print(txt, file =f)
with open (‘’my2.txt’’,’’at’’) as f:
print(txt, file =f)
with open (‘’my2.txt’’,’’r’’) as f:
print(len(f.readlines()))

```
txt = 'abc\ndef'

with open("my2.txt", "wt") as f:
    print(txt, file=f)
with open("my2.txt", "at") as f:
    print(txt, file=f)
with open("my2.txt", "r") as f:
    print(len(f.readlines()))

# что будет выведено на экран?
```

a. 3
b. 12
c. 4
d. 16
e. 2
f. Выполнение ячейки приведет к сообщению об ошибке

Что будет выведено на экран?
txt = ‘abc\ndef’
with open (‘’my2.txt’’,’’wt’’) as f:
print(txt,end =’ ’, file =f)
with open (‘’my2.txt’’,’’at’’) as f:
print(txt, file =f)
with open (‘’my2.txt’’,’’r’’) as f:
print(len(f.readlines()))

```
txt = 'abc\ndef'

with open("my2.txt", "wt") as f:
    print(txt, end='', file=f)
with open("my2.txt", "at") as f:
    print(txt, file=f)
with open("my2.txt", "r") as f:
    print(len(f.readlines()))

# что будет выведено на экран?
```

a. 16
b. Выполнение ячейки приведет к сообщению об ошибке
c. 4
d. 12
e. 2
f. 3

Файл my.txt содержит текст. Что будет выведено на экран?

```
#файл my.txt содержит текст:
txt = 's1\ns2\ns3\ns4'

with open("my.txt", "w") as f:
    print(txt, file=f)
with open("my.txt", "r") as f:
    fl = sum(len(line) for line in f)

# что будет выведено на экран?
fl
```

a. 8
b. 14
c. 16
d. 13
e. 12
f. Выполнение ячейки приведет к сообщению об ошибке

Что будет выведено на экран?

```
txt = 'abc\ndef'

with open("my2.txt", "wt") as f:
    print(txt, end='', file=f)
with open("my2.txt", "wt") as f:
    print(txt, file=f)
with open("my2.txt", "r") as f:
    print(len(f.readlines()))

# что будет выведено на экран?
```

a. 12
b. 16
c. Выполнение ячейки приведет к сообщению об ошибке
d. 3
e. 4
f. 2

Файл my.txt содержит текст. Чем завершится выполнение ячейки?

```
#файл my.txt содержит текст:
# s1\ns2\ns3\ns4\n
with open("my.txt", "r") as f:
    print(next(f))
for line in f:
    print(line)
# чем завершится выполнение ячейки?
```

a. выводом на экран 3х строк
b. выводом на экран 1й строки
c. выводом на экран 5и строк
d. выводом на экран 4х строк
e. Сообщение об ошибке (обращение к несуществующей переменной f)
f. Сообщение об ошибке (выполнение операции с закрытым файлом)

Файл my.txt содержит текст. Чем завершится выполнение ячейки?

```
#файл my.txt содержит текст:
txt = 's1\ns2\ns3\ns4'

with open("my.txt", "w") as f:
    print(txt, file=f)
with open("my.txt", "rb") as f:
    fl = sum(len(line) for line in f)

# что будет выведено на экран?
fl
```

a. 12
b. 8
c. 13
d. Выполнение ячейки приведёт к сообщению об ошибке
e. 14
f. 16

Семестр II

ООП-1

Для чего используются конструкторы?
Выберите один ответ:
a. для удаления объекта
b. для создания подкласса
c. для связи объекта с классом
d. для создания методов классов
e. для формирования иерархии классов
f. для инициализации объекта

Чего не дает абстракция:
Выберите один ответ:
a. разделения кода на сильносвязные блоки
b. снижения связности внутри кода, реализующего абстракцию
c. упрощения программного интерфейса
d. повышения структурированности программного кода
e. снижения связности крупных блоков в программном коде
f. уменьшения количества классов

Утиная типизация в Python позволяет:
a. создавать объект, являющийся одновременно представителем нескольких классов
b. делать классы-наследники несвязанными с классами-родителями
c. без использования механизма наследования наследовать реализацию методов
d. равноправно использовать представителей неродственных классов если эти классы имеют одинаковый интерфейс
e. менять тип объекта во время его существования
f. удалить атрибут базового класса

Полиморфизм позволяет:
a. скрывать реализацию от пользователя класса
b. изменять набор методов в дочерних классах
c. ограничевать доступ к некоторым переменным и методам
d. организовывать иерархию наследования
e. работать с различными реализациями классов через один интерфейс
f. разным объектам одного класса иметь разное поведение

Что нельзя сделать при наследовании класса:
Выберите один ответ:
a. создать приватную переменную
b. реализовать новый конструктор
c. создать новый класс
d. удалить атрибут базового класса
e. создать защищенную переменную
f. добавить новый атрибут

Метод super() используется для:
a. доступа к реализациям методов в родительском классе
b. связи родительского и дочернего класса
c. доступа к другим дочерним классам родительского класса
d. доступа из объекта к переменным и методам класса этого объекта
e. маркирования качественно реализованного класса
f. получения значениям переменных в объектах родительского класса

Функциональное программирование 1

Чем является функция f?

```
def f(x):
    print(x)
    if x > 0:
        return f(x-1) + f(x-2)
    else:
        return x
# Чем являяется фунция f?
```

Выберите один ответ:
a. Функцией без побочных эффектов
b. Функцией высшего порядка
c. Замыканием
d. Лямбда-функцией
e. Функцией, написанной с ошибкой
f. Рекурсивной функцией

#Декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
Каков результат вы
зова функции:

```
import functools

def ag_arg(function):
# декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
    @functools.wraps(function)
    def wrapper(*args, **kwargs):
        return function(*[abs(a) for a in args], **kwargs)
    return wrapper

@ag_arg
def g(a, b, c, d):
    return a + b + c + d

# Каков результат вызова функции:
g(-1,2,d=5,c=-3)
```

Выберите один ответ:
a. 3
b. 9
c. 11
d. Ошибка
e. 5
f. 8

#Чем является функция f?

```
abc = 10
def f(g):
    return g(abc
# Чем является фунция f?
```

Выберите один ответ:

a. Замыканием

b. Функцией, написанной с ошибкой

c. Функцией без побочных эффектов

d. Рекурсивной функцией

e. Функцией высшего порядка

f. Чистой функцией

#Декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
Каков результат вызова функции:

```
import functools

def af_arg(function):
# декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
    @functools.wraps(function)
    def wrapper(*args, **kwargs):
        return function(*[abs(a) for a in args], **{k:abs(v) for k, v in kwargs.items()})
    return wrapper

@af_arg
def f(a, *b):
    return a + sum(b)

# Каков результат вызова функции:
f(-1,1,-2)
```

Выберите один ответ:

a. 4

b. 0

c. 2

d. -3

e. -2

f. Ошибка

#Чем является функция f?

```
def f(a):
    def g(b):
        c = a + b
        print(c)
        return c
    return g
# Чем является фунция f?
```

Выберите один ответ:

a. Замыканием
b. Функцией, написанной с ошибкой
c. Лямбда-функцией
d. Чистой функцией
e. Функцией без побочных эффектов
f. Рекурсивной функцией

Декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
Каков результат вызова функции:

```
import functools

def ah_arg(function):
# декоратор, который обеспечивает, что wrapper будет носить имя исходной функции и ее строку документирования.
    @functools.wraps(function)
    def wrapper(*args, **kwargs):
        print(kwargs)
        return function(*args, **{k:abs(v) for k, v in kwargs})
    return wrapper

@ah_arg
def h(a, b, c, d):
    return a + b + c + d

# Каков результат вызова функции:
h(2, -3, d=5, c=-1)
```

Выберите один ответ:
a. 3
b. 11
c. 10
d. 5
e. 9
f. Ошибка

Функциональное программирование 2

list(map(lambda *a: sum(a), range(4), range(4,8), range(9,13)))

```
list(map(lambda *a: sum(a), range(4), range(4,8), range(9,13)))
```

Выберите один ответ:

a. [12, 15, 18, 21]

b. Сообщение об ошибке

c. [6, 22, 42]

d. [13, 16, 19, 22]

e. 70

f. [13, 16, 19]

import functools as ft

ft.reduce(lambda n, s: n+(1 if ‘is’ in s else 0), ‘This is a test’.split(), 0)

```
import functools as ft
ft.reduce(lambda n, s: n + (1 if 'is' in s else 0) , 'This is a test'.split(), 0)
```

Выберите один ответ:

a. 2

b. [1, 1, 0, 0]

c. 1

d. Сообщение об ошибке

e. '1100'

f. '01100'

import itertools as itl

list(itl.islice(itl.chain(itl.islice(itl.count(), 3), itl.repeat(1)), 12))

```
import itertools as itl
list(itl.islice(itl.chain(itl.islice(itl.count(), 3), itl.repeat(1)), 12))
```

Выберите один ответ:

a. [0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1]

b. [0, 1, 1, 1, 2, 1, 3, 1, 4, 1, 5, 1, 6, 1, 7, 1, 8, 1, 9, 1, 10, 1, 11, 1]

c. [0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

d. [0, 1, 2, 1, 1, 1, 0, 1, 2, 1, 1, 1]

e. [0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]

f. [0, 1, 2, 1, 1, 1, 3, 4, 5, 1, 1, 1]

Каков результат вызова `list(super_iterator(range(8),[10,11]))`:

```
def super_iterator(a, b):
    ai = iter(a)
    bi = iter(b)
    try:
        while True:
            yield next(ai)
            yield next(ai)
            yield next(bi)
    except StopIteration:
        pass
```

Выберите один ответ:
a. [0, 1, 10, 2, 3, 11]
b. [0,10]
c. [0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 10, 11]
d. Сообщение об ошибке
e. [0, 1, 10, 2, 3, 11, 4, 5, 6, 7]
f. [0, 1, 10, 2, 3, 11, 4, 5]

Каков результат вызова: list(super_iterator([1,2,3,4],[10,11]))

```
def super_iterator(a, b):
    ai = iter(a)
    bi = iter(b)
    try:
        while True:
            yield next(ai)
            yield next(bi)
    except StopIteration:
        pass
```

Выберите один ответ:
a. [1, 10]
b. [1, 2, 3, 4, 10, 11]
c. [1, 10, 2, 11, 3]
d. [1, 10, 2, 11, 3, 4]
e. Сообщение об ошибке
f. [1, 10, 2, 11, 3, 11, 4]

import itertools as itl
list(itl.islice(itl.chain(itl.zip_longest(itl.count(10, 10), itl.repeat('1')), 5))

```
import itertools as itl
list(itl.islice(itl.zip_longest(itl.count(10,10), itl.repeat('1')), 5))
```

Выберите один ответ:

a. [(10, '1'), (10, '1'), (10, '1'), (10, '1'), (10, '1')]
b. [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, '1', '1', '1', '1', '1']
c. [(10, '1'), (20, '1'), (30, '1'), (40, '1'), (50, '1'), (60, None), (70, None), (80, None), (90, None), (100, None)]
d. [10, 20, 30, 40, 50, '1', '1', '1', '1', '1']
e. [(10, '1'), (20, '1'), (30, '1'), (40, '1'), (50, '1'), (60, '1'), (70, '1'), (80, '1'), (90, '1'), (100, '1')]
f. [(10, '1'), (20, '1'), (30, '1'), (40, '1'), (50, '1')]

Алгоритмы 1

Имеется асимптотическая оценка сложности алгоритма в O(ln n) операций. Эта оценка гарантирует, что количество операций при выполненении алгоритма:
Выберите один ответ:
a. Меньше или равна C ln n, при боьших n.
b. Меньше или равна C ln n, при любых n.
c. Не превышает C ln n, где C константа, большая 0, при n не больше определенного значения.
d. Не превышает C ln n, при боьших n, где C константа, большая 0.
e. Всегда не превышает C ln n, где C константа, большая 0.
f. Не превышает ln n при больших n .

Какое из представленных ниже утверждений относительно динамического массива (ДМ) не верно:
Выберите один ответ:
a. Добавление элемента в конец ДМ всегда требует O(1) операций.
b. Использование ДМ создает накладные расходы по памяти.
c. Удаление элмента из середины ДМ сложнее, чем получение этого элемента по индексу.
d. Добавление элмемента в начало ДМ сложнее, чем добавление в конец ДМ.
e. Получение значения элемента ДМ по его индексу треубет O(1) операций.
f. Проверка вхождения значения в ДМ требует O(1) операций.

#Что содержится в очереди в результате выполненных действий?

```
q = Queue()
q.enqueue(10)
q.enqueue(8)
q.enqueue(6)
q.dequeue()
q.enqueue(4)
q.dequeue()
q.first()
#Что содержится в очереди в результате выполеныных действий?
```

Выберите один ответ:
a. Queue([6])
b. Queue([10])
c. Queue([6, 4])
d. Queue([4, 6])
e. Queue([10, 8])

f. Queue([4])

#Что содержится в стеке в результате выполненных действий?

```
s = Stack()
s.push(10)
s.push(8)
s.push(6)
s.pop()
s.push(4)
s.pop()
s.top()
# Что содержится в стеке в результате выполеныных действий?
```

Выберите один ответ:

a. Stack([10, 8])

b. Stack([4])

c. Stack([8, 10])

d. Stack([10])

e. Stack([8])

f. Stack([6, 4])

Какое из представленных ниже утверждений относительно динамического массива (ДМ) и связанного списка (СС) не верно:

Выберите один ответ:

a. Добавление большого количества элементов в конец длинного ДМ потребует больше операций, чем для длинного СС.

b. Операция вставки в начало ДМ медленнее чем в СС.

c. Операция вставки в середину коллекции в СС осуществляется быстрее, чем в ДМ.

d. Операция проверки вхождения элемента для СС и ДМ имеет сопоставимую сложность.

e. СС может быть организован так, что вставка в начало и в конец списка будут иметь сопоставимую сложность.

f. Доступ к середине списка в ДМ медленнее, чем в СС.

Что содержится в двусторонней очереди в результате выполненных действий ?

```
q = deque()
q.append(10)
q.append(8)
q.appendleft(6)
q.pop()
q.append(4)
q.popleft()
#Что содержится в двусторонней очереди в результате выполеныных действий?
```

Выберите один ответ:

a. deque([4, 8])

b. deque([8, 4])

c. deque([6, 10])

d. deque([4, 10])

e. deque([10, 4])

f. deque([10, 6])

Алгоритмы 2

Последовательная реализация поиска используется для отсортированного (a_rnd) и не отсортированного массива (a_sort). Какое из приведенных ниже утрверждений не верно?

Выберите один ответ:

a. Удвоение длинны массива удовит среднее время поиска в a_rnd.

b. Скорость роста сложности поиска от длины массива для a_rnd и a_sort одинакова.

c. Удвоение длинны массива удовит среднее время поиска в a_sort.

d. Среднее время поиска элемнта, отсутствующего в массиве, для a_sort меньше чем для a_rnd.

e. Среднее время поиска элемнта, имеющегося в массиве, для a_sort меньше чем для a_rnd.

f. Использование a_sort дает преимущества перед a_rnd.

Какая из сортировок тербует определения шага, который используется для формирования серий, внутри которых каждое значение отстоит от своих соседей на заданную длину шага?

Выберите один ответ:

a. Сортировка слиянием

b. Сортировка выбором

c. Сортировка включением

d. Обменная сортировка

e. Быстрая сортировка

f. Сортировка Шелла

При эффективной реализации поиска в отсортированном массиве какой эффект на усредненный (математическое ожидаение) объем вычислений, необхдимых для обнаружения искомого элемента, приведет удвоение размера массива:

Выберите один ответ:

a. Среднее количество шагов поиска увеличится в ln 2 раз.

b. Среднее количество шагов поиска увеличится на 1 операцию поиска.

c. Среднее количество шагов поиска не поменяется.

d. Среднее количество шагов поиска увеличится на ln 2 операций поиска.

e. Среднее количество шагов поиска увеличится в sqrt 2 раз.

f. Среднее количество шагов поиска удвоится.

При помощи обменной сортировки выполняется сортировка по возрастанию. В сортировке используется оптимизация, останавливающая сортировку отсортированного массива. Какой из представленных списков потребует наибольшего количества операций?

Выберите один ответ:

a. [4, 5, 6, 1, 2, 3]
b. [1, 2, 3, 4, 5, 6]
c. [5, 6, 3, 4, 1, 2]
d. [6, 5, 3, 4, 2, 1]
e. [6, 2, 3, 4, 5, 6]
f. [4, 1, 5, 3, 2, 6]

При эффективной реализации поиска в отсортированном массиве какой эффект на усредненный (математическое ожидаение) объем вычислений, необхдимых для обнаружения искомого элемента, приведет удвоение размера массива
Выберите один ответ:
a. Среднее количество шагов поиска увеличится в ln 2 раз.
b. Среднее количество шагов поиска удвоится.
c. Среднее количество шагов поиска увеличится в sqrt 2 раз.
d. Среднее количество шагов поиска не поменяется.
e. Среднее количество шагов поиска увеличится на ln 2 операций поиска.
f. Среднее количество шагов поиска увеличится на 1 операцию поиска.

Какая из сортировок требует определения значения на основании которого массив разделяется на два подмассива, для которых выполняется некоторое свойство относительно этого числа?
Выберите один ответ:
a. Сортировка Шелла
b. Быстрая сортировка
c. Сортировка выбором
d. Сортировка слиянием
e. Обменная сортировка
f. Сортировка включением

Какая из сортировок требует разбиения массива в виде иерархии из пар массивов половинной длины?
Выберите один ответ:
a. Сортировка включением
b. Сортировка слиянием
c. Обменная сортировка
d. Быстрая сортировка
e. Сортировка Шелла
f. Сортировка выбором

Алгоритмы 3

Какой(какие) из порядков обхода двоичных деревьев (содержащих > 1 узла) не может вернуть в качестве последнего узла лист дерева? Выбрать наиболее точный ответ.
Выберите один ответ:
a. Прямой порядок обхода дерева (кроме случая, когда двоичное дерево вырождено в список)
b. Обратный порядок обхода дерева
c. Симметричный порядок обхода дерева
d. Прямой, симметричный и обратный порядок обхода дерева
e. Прямой, и обратный порядок обхода дерева
f. Прямой порядок обхода дерева

Какой(какие) из порядков обхода двоичных деревьев (содержащих > 1 узла) может вернуть в качестве крайнего (первого или последнего) узла узел, не являющихся ни листом ни корнем дерева? Выбрать наиболее точный ответ.
Выберите один ответ:
a. Ни один из рассмотренных порядков обхода
b. Прямой, симметричный и обратный порядок обхода дерева
c. Прямой порядок обхода дерева (кроме случая, когда двоичное дерево вырождено в список)
d. Прямой порядок обхода дерева
e. Обратный порядок обхода дерева
f. Симметричный порядок обхода дерева

Дан список, представляющий собой линейную запись двоичной кучи (элемент с индеком 0 не является члеом кучи): [None, 3, 8, 4, 9, 16, 8, 5, 10, 18, 17, 32]. Какое значения у элемента, являющегося самым правым элементом уровня 2 в куче?
Выберите один ответ:
a. 4
b. 5
c. 17
d. 9
e. 10
f. 8

Дан список, представляющий собой линейную запись двоичной кучи (элемент с индеком 0 не является члеом кучи): [None, 6, 16, 8, 18, 32, 16, 10, 20, 36, 34]. Какое значения у элемента, являющегося самым левым листом в куче?

Выберите один ответ:
a. 34
b. 36
c. 6
d. 10
e. 18
f. 20

Какой(какие) из порядков обхода двоичных деревьев (содержащих > 1 узла) может вернуть в качестве крайнего (первого или последнего) узла корень дерева? Выбрать наиболее точный ответ.
Выберите один ответ:
a. Прямой, и обратный порядок обхода дерева
b. Прямой порядок обхода дерева
c. Прямой порядок обхода дерева (кроме случая, когда двоичное дерево вырождено в список)
d. Обратный порядок обхода дерева
e. Прямой, симметричный и обратный порядок обхода дерева
f. Симметричный порядок обхода дерева

Дан список, представляющий собой линейную запись двоичной кучи (элемент с индеком 0 не является члеом кучи): [None, 12, 32, 16, 36, 64, 32, 20, 40, 72]. Какое значения у элемента, являющегося самым левым элементом уровня 2 в куче?
Выберите один ответ:
a. 32
b. 36
c. 40
d. 16
e. 20
f. 64

Алгоритмы 4

a и b - относятся к одному из встроенных типов Python. Для них выражение: hash(a) == hash(b) возвращает значение `True`.
Выберите НЕверное утрверждение о a и b.

Выберите один ответ:
a. Значение len(set([a, b])) может равняться 2.
b. Значения id(a) и id(b) могут различаться.
c. Верно, что: a == b
d. И a и b относятся к неизменяемым типам.
e. hash(a-b) может не равняться 0.
f. Выражение a > b может иметь значение True.

Выберите НЕверное утрверждение о: разрешении коллизий при помощи цепочек.
Выберите один ответ:
a. Для данного метода не нужна специализированная хэш-функция.
b. В случае коллизии время вставки по новому ключу больше чем без коллизии.
c. Требует дополнительной памяти в случае коллизии.
d. Цепочка может превышать длину ln m.
e. Не позволяет хранить в таблице значений больше чем имеется слотов.
f. Требует просмотра связанного списка при вставке значения по ключу вызвавшему коллизию.

Выберите НЕверное утрверждение о: разрешении разрешение коллизий при помощи открытой адресации.
Выберите один ответ:
a. Требует дополнительной памяти в случае коллизии.
b. Для данного метода нужна специализированная хэш-функция.
c. В случае коллизии время вставки по новому ключу больше чем без коллизии.
d. При использовании равного объема затрачиваемой памяти позволяет снизить колчество коллизий по сравнению с методом цепочек.
e. Не позволяет хранить в таблцие значений больше чем имеется слотов.
f. Значение по фиксированному ключу может оказаться в любом из слотов таблицы.

Выберите НЕверное утрверждение о таблице с прямой адресацией.

Выберите один ответ:

a. Требует не больше ячеек чем имеется различных значений ключа.
b. По сравнению с хэш-таблицей обеспечивает меньшую скорость доступа к данным в наихудшем случае.
c. Может работать не только с целочисленными ключами.
d. В одной ячейке не может содержать значения более чем по одному ключу.
e. Менее эффективна чем хэш-таблица в смысле затрат памяти.
f. Обеспечивает предсказуемую скорость доступа к значениям по всем ключам.