Государство Израиль
Министерство просвещения
Тип экзамена: на аттестат зрелости
Время проведения экзамена: лето 2023 года
Номер вопросника: 899381

מדינת ישראל
משרד החינוך
סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ג, 2023
מספר השאלון: 899381

Компьютеры

מ ד ע י ה מ ח ש ב

Указания

а. Продолжительность экзамена: три часа.

б. Строение вопросника и ключ к оценке:
в этом вопроснике три раздела.

Раздел первый:	(1 × 10) + (1 × 15)	–	25 баллов
Раздел второй:	(2 × 25)	–	50 баллов
Раздел третий:	(1 × 25)	–	25 баллов
	Всего	–	100 баллов

в. Разрешенный вспомогательный материал:
любой вспомогательный материал, за исключением калькулятора с возможностью программирования.

г. Особые указания:

(1) **На внешней обложке** тетради **укажите название курса**, который **вы изучали**.
Курс – это один из трех следующих курсов:
"Алгоритмы", "Вычислительные модели",
"Объектно-ориентированное программирование".

(2) Все программы, которые вы должны написать на языке программирования в первом и втором разделах, пишите только на одном языке – Java или C# .

Примечание: баллы сниматься не будут, если в программах вы напишете большую букву вместо маленькой или наоборот.

הוראות

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שלושה פרקים.

פרק ראשון	(1×10) + (1×15)	25 נקודות
פרק שני	(2×25)	50 נקודות
פרק שלישי	(1×25)	25 נקודות
סך הכול		100 נקודות

ג. חומר עזר מותר בשימוש:
כל חומר עזר, חוץ ממחשבון שיש בו אפשרות תכנות.

ד. הוראות מיוחדות:

(1) **רשמו על הכריכה החיצונית** של המחברת את **שם המסלול שלמדתם**.
המסלול הוא אחד שלושת המסלולים האלה: אלגוריתמים, מודלים חישוביים, תכנות מונחה עצמים.

(2) את כל התוכניות שיש לכתוב בשפת מחשב בפרקים הראשון והשני כִּתבו בשפה אחת בלבד – Java או C# .

הערה: לא יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

יש לכתוב במחברת הבחינה בלבד. יש לרשום "טיוטה" בראש כל עמוד המשמש טיוטה.
כתיבת טיוטה בדפים שאינם במחברת הבחינה עלולה לגרום לפסילת הבחינה.
Пишите только в экзаменационной тетради. Напишите слово «טיוטה» в начале каждой страницы, отведенной вами под черновик. Выполнение любых черновых записей на листах, не относящихся к экзаменационной тетради, может привести к тому, что экзамен будет аннулирован.

Желаем успеха!

ב ה צ ל ח ה !

В о п р о с ы

В этом вопроснике три раздела.

Вы должны ответить на вопросы трех этих разделов, согласно инструкциям в каждом разделе.

Примечание: в каждом вопросе, в котором требуется ввод данных,
нет необходимости проверять их корректность.

Для решающих на языке Java: в каждом вопросе, в котором требуется ввод данных,
считайте, что в программе написана команда:

```
Scanner input = new Scanner (System.in);
```

РАЗДЕЛ ПЕРВЫЙ (25 баллов)

Ответьте на вопрос 1 – обязательный (10 баллов).

1. "Массив, упорядоченный по положительным числам" [מערך ממוין בחיוביים] – это массив целых чисел, в котором все числа, большие, чем 0 , расположены в порядке возрастания (порядок возрастания – каждое число больше предыдущего положительного числа или равно ему).

Пример:

Массив arr , приведенный ниже – это "массив, упорядоченный по положительным числам":

	0	1	2	3	4	5	6	7	8
arr	**5**	**9**	− 3	**17**	0	**29**	− 20	− 40	**29**

Объяснение: положительные числа в массиве (5 , 9 , 17 , 29 , 29) расположены в порядке возрастания.

Напишите внешний метод [פעולה חיצונית], с заголовком posOrder на языке Java или с заголовком PosOrder на языке C#, который получает массив целых чисел – arr .

Метод вернет true, если arr является "массивом, упорядоченным по положительным числам", иначе метод вернет false.

Считайте, что в этом массиве есть по меньшей мере два положительных числа.

/продолжение на странице 3/

Ответьте на один из вопросов 2–3 (15 баллов).

2. "Массив разностей" – это массив целых чисел, в котором разность чисел в каждых двух соседних ячейках постоянна.

Пример массива разностей:

0	1	2	3	4	5
5	9	13	17	21	25

Объяснение: разность чисел в каждых двух соседних ячейках этого массива равна 4 .

Реализуйте приведенный ниже внешний метод [פעולה חיצונית]:

Java – public static int missingNum (int [] arr)

C# – public static int MissingNum (int [] arr)

Метод получает массив разностей – arr , в котором не хватает одной ячейки, и по этой причине в массиве есть две соседние ячейки, разность чисел в которых отличается от постоянной разности. Этот метод вернет число, которое должно быть в недостающей ячейке.

Предположите, что длина полученного массива arr не менее 4 .

Пример:

Для массива arr , приведенного ниже, метод вернет число 10 .

	0	1	2	3	4	5
arr	6	**8**	**12**	14	16	18

Объяснение: разность чисел в двух соседних ячейках равна 2 , кроме ячеек с индексами 1 и 2 , разность чисел в которых равна 4 . Это объясняется тем, что между ними не хватает ячейки с числом 10 .

/продолжение на странице 4/

3. На полосах общественного транспорта установлены камеры, которые фотографируют все проезжающие там машины.

Дан класс **CarInfo** – информация о сфотографированной машине; в этом классе три атрибута [תכונות]:

- id – номер сфотографированной машины, типа строка.
- privateCar – если сфотографированная машина является частной, значение этого атрибута будет true, а если сфотографированная машина является общественным транспортным средством, то false .
- speed – скорость движения сфотографированной машины, целого типа.

Считайте, что определены методы get/Get и set/Set для каждого из атрибутов этого класса.
На машину, которая едет по полосе общественного транспорта, будет выписан штраф, если соблюдается по меньшей мере одно из следующих условий:

– машина является частной
– машина едет с превышением допустимой скорости

(א) Напишите внутренний метод [פעולה פנימית] класса **CarInfo** с заголовком illegal на языке Java или с заголовком Illegal на языке C# , получающий допустимую скорость – maxSpeed, целого типа.
Метод вернет true , если машина нарушила правила дорожного движения (если машина является частной и/или ехала с превышением допустимой скорости), иначе метод вернет false .

Дан класс **CameraInfo** – информация о камере; в этом классе три атрибута:

- city – код города, в котором эта камера установлена, целого типа. Код – число от 0 до 99 (включительно). Есть 100 городов, и у каждого города есть уникальный код.
- maxSpeed – допустимая скорость в районе этой камеры, целого типа.
- cars – массив типа **CarInfo** , содержащий сфотографированные камерой машины (массив без значений null).

Считайте, что определены методы get/Get и set/Set для каждого из атрибутов этого класса.

(ב) **(1)** Напишите внутренний метод [פעולה פנימית] класса **CameraInfo** с заголовком allGood на языке Java или с заголовком AllGood на языке C# , который вернет true , если все сфотографированные камерой машины **не** нарушили правила дорожного движения, иначе метод вернет false .
Можно использовать метод, который вы написали в пункте (א).

(2) Напишите внешний метод [פעולה חיצונית] с заголовком legalCities на языке Java или с заголовком LegalCities на языке C#.
Метод получает массив – cameras типа **CameraInfo** , содержащий информацию камер, которые установлены в 100 различных городах.
Метод вернет количество городов, в которых **не** было обнаружено ни одного нарушения правил дорожного движения.
Примечание: возможно, что есть более одной камеры в одном и том же городе.
Можно использовать метод, который вы написали в пункте (ב)-(1).

/продолжение на странице 5/

РАЗДЕЛ ВТОРОЙ (50 баллов)

Внимание: во всех вопросах, где требуется реализация [מימוש], разрешается пользоваться методами классов Очередь, Стек, Двоичное дерево и Звено [תור, מחסנית, עץ בינרי וחוליה], не реализуя их. Если вы используете дополнительные методы, вы должны их реализовать [לממש].

Ответьте на два из вопросов 4–6 (за каждый вопрос – 25 баллов).

4. В этом вопросе к классу **Queue** добавили метод size/Size , приведенный ниже. Можно использовать данный метод, не реализуя его.

Заголовок метода	Описание метода
Java – public int size() **C#** – public int Size()	Метод возвращает число элементов в очереди [תור].

Реализуйте внешний метод [פעולה חיצונית], приведенный ниже:

Java – public static boolean twoSum (Queue<Integer> q, int x)

C# – public static bool TwoSum (Queue<int> q, int x)

Метод вернет true , если в очереди q , которая будет получена, есть два числа, сумма которых равна значению параметра x . Иначе метод вернет false.

Пример: для очереди q , приведенной ниже, и x = 10 метод вернет true , потому что в этой очереди есть два числа (9 , 1) , сумма которых равна 10 .

	Голова очереди							
q	5	4	**1**	4	3	15	**9**	

Примечание:

– Считайте, что в очереди q есть по меньшей мере два элемента.

– Нет необходимости сохранять очередь q .

– В этом вопросе **запрещено** пользоваться массивом и связным списком [רשימה מקושרת].

/продолжение на странице 6/

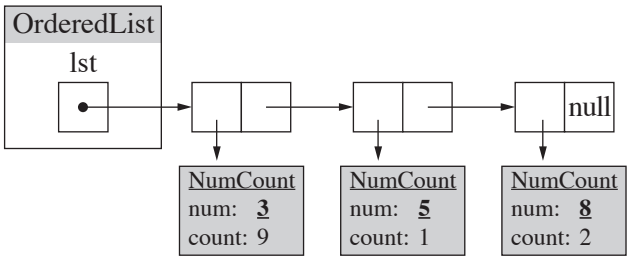
5. Дан класс **NumCount** – число значений; в этом классе два атрибута:

- num – численное значение, целого типа
- count – число (счетчик) появлений этого значения (num), целого типа. Число больше или равно 0 .

Считайте, что определены методы get/Get и set/Set для каждого из атрибутов этого класса и конструктор [פעולה בונה], который получает значения для атрибутов этого класса.
Дан класс **OrderedList** – отсортированная цепочка; и в нем один атрибут:

- lst – указывает на голову цепочки звеньев типа **NumCount .**

Цепочка звеньев отсортирована в порядке возрастания значения атрибута – num . Значение атрибута num различно в каждом звене.
Пример: для цепочки, приведеннй ниже, выполняется условие класса (цепочка отсортирована в порядке возрастания значения атрибута num , и значение атрибута num различно в каждом звене).



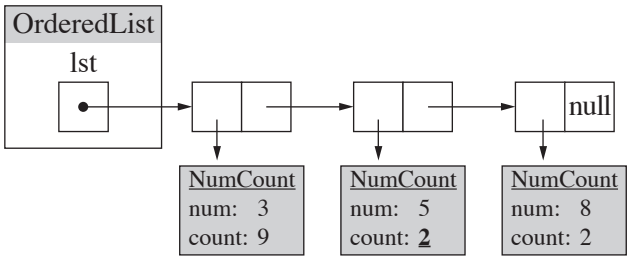
(א) (1) Реализуйте в классе OrderedList внутренний метод [פעולה פנימית], приведенный ниже:

Java – public void insertNum (int x)

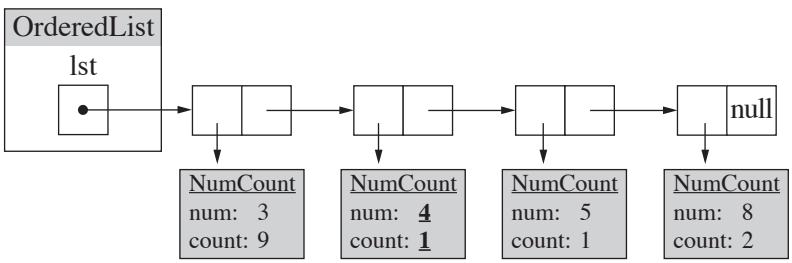
C# – public void InsertNum (int x)

Метод добавляет величину x к списку следующим способом:
– если в цепочке существует звено, атрибут num которого равен x , метод увеличит атрибут count (число появлений) в этом звене на 1.
– если цепочка пустая или в цепочке не существует звена, атрибут num которого равен x , метод добавит новое звено, у которого атрибут num будет равен x и атрибут count будет равен 1 , в месте, сохраняющем порядок возрастания цепочки.

Пример: для приведенной выше цепочки и для x = 5 после вызова метода цепочка будет выглядеть так:



Дополнительный пример: для той же цепочки, представленной выше (в первом примере) и для x = 4 после вызова метода цепочка будет выглядеть так:



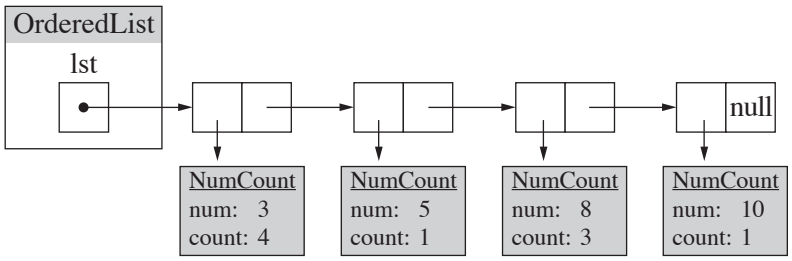
(2) Какова временная сложность выполнения [סיבוכיות זמן הריצה] метода, который вы написали в пункте (א)-(1)? Обоснуйте свой ответ.

Обратите внимание: продолжение вопроса на следующей странице.

/продолжение на странице 7/

(ב) "Значение позиции n " – это значение, появляющееся на месте n по порядку с начала цепочки (при учете всего количества появлений – count каждого значения).

Пример: для приведенной ниже цепочки и для n = 7 метод вернет значение 8 .



Объяснение: последовательный порядок значений этой цепочки в соответствии с количеством их появлений: 3 , 3 , 3 , 3 , 5 , 8 , **8** , 8 , 10 .
n = 7 , и на седьмом месте в последовательности появляется значение 8 . Поэтому метод вернет значение 8 .

Реализуйте в классе OrderedList приведенный ниже внутренний метод [פעולה חיצונית]:
Java – public int valueN (int n)
C# – public int ValueN (int n)

Метод получает число n и возвращает "значение позиции n".
Считайте, что "значение позиции n" существует в цепочке.

/продолжение на странице 8/

6. "Простое число" – число, которое делится только на себя и на 1 (числа 1 и 2 тоже простые).
Ниже приведен внешний метод [הפעולה החיצונית] isPrime/IsPrime . Можно использовать этот метод, не реализуя его.

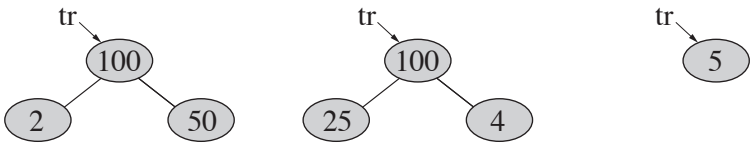
Заголовок метода	Описание метода
Java – public static boolean isPrime (int num) **C#** – public static bool IsPrime (int num)	Метод возвращает true , если полученное значение num – простое число, иначе он возвращает false .

(א) Реализуйте внешний метод, приведенный ниже:

Java – public static boolean addNodes (BinNode<Integer> tr)

C# – public static bool AddNodes (BinNode<int> tr)

Метод получает вершину [צומת] без сыновей [בנים], значение которой больше, чем 0 . Если значение вершины является простым числом, метод возвращает false .
Иначе метод добавляет к вершине двух сыновей, значение произведения [מכפלה] которых равно значению вершины, а значение каждого из них больше, чем 1 .
После добавления метод возвращает true .
Примеры: после вызова этого метода вершины могут выглядеть так
(для значений 5 , 100 , 100):



(ב) Дан метод what/What , принимающий вершину без сыновей, значение которой больше, чем 0 .

На языке **C#**	На языке **Java**

public static void What (BinNode<int> tr)
{
if (AddNodes (tr))
{
What (tr.GetLeft());
What (tr.GetRight());
}
}
	public static void what (BinNode<Integer> tr)
	{
	if (addNodes (tr))
	{
	what (tr.getLeft());
	what (tr.getRight());
	}
	}

(1) Начертите, как будет выглядеть дерево после вызова метода what/What для вершины без сыновей – tr , значение которого 150 .
Необходимо показать трассировку [מעקב].

(2) Объясните, что выполняет метод what/What .

/продолжение на странице 9/

РАЗДЕЛ ТРЕТИЙ (25 баллов)

В этом разделе вопросы из трех разных курсов:

– Алгоритмы [אלגוריתמים], стр. 9–10.
– Вычислительные модели [מודלים חישוביים], стр. 11–12.
– Объектно-ориентированное программирование [תכנות מונחה עצמים] на языке Java, стр. 14–17; объектно-ориентированное программирование на языке C#, стр. 18–21.

Ответьте на один вопрос курса, который вы изучали.

Алгоритмы [אלגוריתמים]

Если вы изучали этот курс, ответьте на один из вопросов 7–8 (25 баллов).

7. В этом вопросе два пункта (א)–(ב). Между ними нет связи. Ответьте на вопросы обоих пунктов.

(א) Ниже приведены пять утверждений. Выберите четыре из них.

Для каждого из выбранных вами утверждений напишите в своей тетради его номер и укажите, верно оно или неверно.

Если вы указали, что утверждение верно, обоснуйте почему, а если вы указали, что утверждение неверно, приведите опровергающий пример или обоснуйте почему.

1. Для каждого покрывающего дерева [עץ פורש] DFS одного и того же неориентированного [לא מכוון] графа G всегда есть одинаковое число листьев [עלים].

2. Высота покрывающего дерева BFS неориентированного [לא מכוון] графа G всегда меньше или равна высоте покрывающего дерева DFS того же графа.

3. Неориентированный [לא מכוון] граф G с n вершинами и n–2 ребрами всегда является несвязным [לא קשיר].

4. В каждом ориентированном [מכוון] графе G , у которого n вершин, m ребер и $m >= n$, существует цикл.

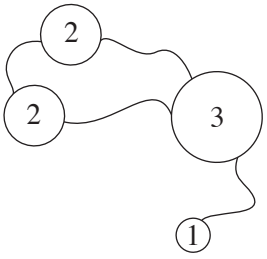
5. Неориентированный граф G , у которого степени [דרגות] всех вершин больше, чем 0 , всегда является связным [קשיר].

(ב) "Максимальное покрывающее дерево" – это покрывающее дерево [עץ פורש] неориентированного [לא מכוון] графа G , у которого сумма значений ребер является максимальной.

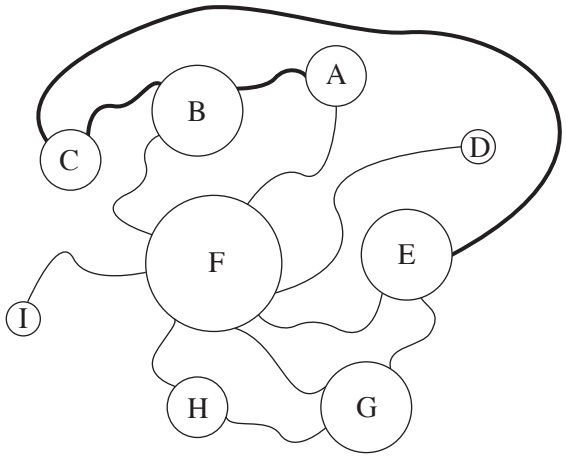
Напишите алгоритм, который находит в графе G "максимальное покрывающее дерево".

/продолжение на странице 10/

8. В "сетке улиц" переход с улицы на улицу всегда проходит через площадь.
Величина (в метрах) радиуса площади соответствует числу улиц, которые она соединяет.
Например, радиус площади, соединяющей 3 улицы, равен 3 метрам, а радиус площади, соединяющей 6 улиц, равен 6 метрам.
Пример: ниже приведен чертеж сетки улиц. Внутри каждой площади обозначен радиус этой площади.



Перед вами сетка улиц NET одного города:



Пешеход должен пройти от одной площади до другой самым коротким маршрутом. Самый короткий маршрут – это маршрут, в котором сумма радиусов всех площадей, через которые он проходит, самая маленькая. Сумма радиусов не включает радиус площади, с которой он начинает свой маршрут, но включает радиус площади, на которой он завершает свой маршрут.
Пример: на приведенном выше чертеже сетки улиц NET выделен самый короткий маршрут от площади A до площади E .
Величина радиусов на этом маршруте: 3+2+3 , а их сумма равна 8 . Эта сумма является наименьшей из всех возможных.

(א) Каков самый короткий маршрут от площади H до площади C в сетке улиц NET ?
Напишите названия площадей этого маршрута по порядку слева направо
(нет необходимости в трассировке [מעקב]).

(ב) **(1)** Напишите алгоритм, находящий для любой сетки улиц самый короткий маршрут от площади K_1 до площади K_2 .
Примечание: необходимо написать эффективный алгоритм, который не проходит через все возможные маршруты.

(2) Начертите граф, представляющий сетку улиц NET , приведенную выше, таким способом, который будет соответствовать написанному вами алгоритму.

/продолжение на странице 11/

Вычислительные модели [מודלים חישוביים]

Если вы изучали этот курс, ответьте на один из вопросов 9–10 (25 баллов).

9. Даны языки $L_1 - L_4$ над алфавитом {a,b} :

L_1 = язык всех слов, в которых число появлений буквы a равно числу появлений буквы b .

L_2 = язык всех слов, в которых число появлений буквы a больше числа появлений буквы b .

L_3 = язык всех слов, в которых есть более трех букв.

L_4 = язык всех слов, которые начинаются с буквы a и заканчиваются буквой b или начинаются с буквы b и заканчиваются буквой a .

Ответьте на все вопросы пунктов (א)– (ז):

(א) Постройте детерминированный конечный [סופי דטרמיניסטי] автомат, принимающий язык L_4 .

(ב) Докажите, что язык $L_3 \cap L_4$ является регулярным.

(ג) Напишите язык, получаемый действием $L_1 \cup L_2$. Является ли получаемый язык регулярным? Обоснуйте свой ответ.

(ד) Напишите язык, получаемый действием $L_1 \cup L_2 \cup \overline{L}_2$.

(ה) Является ли язык $L_1 \cap L_2$ регулярным? Обоснуйте свой ответ.

(ו) Постройте неполный детерминированный конечный автомат, принимающий язык $L_2 \cap \overline{L}_3$.

(ז) Напишите язык, получаемый действием $L_4 \cap R(L_4)$.

/продолжение на странице 12/

10. Метод **Generate** производится над двумя числами, длина которых одинакова и которые содержат только цифры 0 и 1 . Метод создает новое слово такой же длины следующим способом: если у обоих чисел цифра на месте n равна 0 , то буква на месте n будет F , иначе буква на месте n будет T .

Пример: метод **Generate** на числах x = 11001 и y = 10010 создает слово TTFTT , как показано ниже:

	0	1	2	3	4
x	1	1	0	0	1
y	1	0	0	1	0
	Generate				
Получаемое слово	T	T	F	T	T

Постройте машину Тьюринга, которая примет в качестве ввода [קלט] в начале ленты два числа одинаковой длины, содержащие только цифры 0 и 1 .

Машина выведет слово, получаемое методом **Generate** на этих числах.

Указания:

– Два числа при вводе разделены знаком # .

– Не нужно сохранять ввод (можно изменять ввод на различные знаки).

– Новое слово появится в любом месте ленты между двумя знаками $.

– Не нужно проверять корректность ввода.

Пример

Лента памяти до выполнения:

	0	1	2	3	4	5		0	1	2	3	4	5			
⊢	1	0	1	0	1	0	#	1	0	1	1	0	0	△	△	...

Лента памяти после выполнения:

...	$	T	F	T	T	T	F	$	...

/продолжение на странице 13/

Обратите внимание: продолжение экзамена на следующей странице.

/продолжение на странице 14/

Объектно-ориентированное программирование [תכנות מונחה עצמים] на языке Java

Если вы изучали этот курс и пишете на языке Java, ответьте на один из вопросов 11–12 (25 баллов).

11. В ресторане "На уровне" можно заказать место для одного или более посетителей. В конце трапезы посетители платят по отдельности, каждый за себя. Каждый посетитель может выбрать оплату наличными, или в приложении, или кредитной картой. При оплате кредитной картой есть возможность разделить счет на несколько равных платежей.
Для управления платежной системой для ресторана разрабатывают проект с интерфейсом **IPayment** (Interface) и классами **Cash, App, Credit, Reservation, Restaurant**, которые объясняются ниже.

- **IPayment** – платеж
  в интерфейсе есть метод: double getPrice()
- **Cash** – оплата наличными
  Атрибуты класса:
  - sumCash – сумма для оплаты наличными, типа действительное число.
  - name – имя посетителя, типа строка.
- **App** – оплата в приложении
  Атрибуты класса:
  - sumApp – сумма для оплаты в приложении, типа действительное число.
  - phoneNumber – номер телефона, типа строка.
- **Credit** – оплата кредитной картой
  Атрибуты класса:
  - num – число платежей, целого типа.
  - part – сумма каждого платежа, типа действительное число.
  - creditNumber – номер кредитной карточки, типа строка.
- **Reservation** – заказ
  Атрибуты класса:
  - date – дата заказа, типа строка.
  - total – суммарный счет всех посетителей вместе в рамках этого заказа, типа действительное число.
  - payments – массив, в котором хранится платеж каждого из посетителей, типа **IPayment** (массив без значений null).
- **Restaurant** – ресторан
  Атрибуты класса:
  - array – массив длиной 10000 , типа **Reservation** .
  - current – количество сделанных заказов, целого типа.

Считайте, что заказы хранятся последовательно в массиве и среди них нет значений null.

Примечание: считайте, что в каждом классе определены методы get и set и конструкторы.

Обратите внимание: продолжение вопроса на следующей странице.

/продолжение на странице 15/

(א) **(1)** Начертите схему иерархии, описывающую связь между классами и интерфейсом проекта.

Обозначьте реализацию интерфейса посредством стрелки  ,

а включение – посредством символа  .

(2) Ресторан делает скидки посетителям в соответствии с типом оплаты: посетитель, который платит наличными и сумма платежа которого выше 200 шекелей, имеет право на 10% скидки, посетитель, который платит в приложении, всегда получает 5% скидки, а посетитель, который пользуется кредитной картой, платит полную цену.

Метод интерфейса getPrice возвращает общий платеж, который платит клиент после учета скидки (если скидки нет, метод возвращает полную цену). Реализуйте метод интерфейса getPrice в каждом из требуемых классов.

(ב) Напишите в классе **Reservation** метод с заголовком cashTotal. Метод возвращает сумму наличных денег, которую получают при оплате заказа (после учета скидки).

(ג) Ниже приведен метод в классе **Reservation**:

```
public void printDetails () {
    for (int i = 0; i < payments.length; i ++) {
        System.out.println (payments[i].getDetails());
    }
}
```

Этот метод выводит для каждого посетителя определенные данные по следующим критериям:

– если посетитель заплатил наличными, метод выводит его имя.

– если посетитель заплатил в приложении, метод выводит номер его телефона.

– если посетитель заплатил кредитной картой, метод выводит номер его кредитной карты.

Добавьте в проект методы, необходимые для того, чтобы этот метод выполнял требуемое.

Укажите для каждого метода, который вы добавили, к какому классу или к какому интерфейсу он принадлежит.

Указание: запрещено изменять метод printDetails .

/продолжение на странице 16/

12. Даны классы – First и Second:

```
public class First
{
    private static int count = 0;
    protected int x;
    protected int y;
    public First (int num) {
        this.x = num;
        this.y = num;
        count ++;
        System.out.println ("First 1");
    }
    public First (int num1, int num2) {
        this.x = num1;
        this.y = num2;
        count ++;
        System.out.println ("First 2");
    }
    public static int getCount() {
        return count;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public int sum() {
        return this.x + this.y;
    }
    public void add(First other) {
        this.x += other.x;
        this.y += other.y;
        System.out.println("x = "+ this.x +
                           "y = "+ this.y);
    }
}
```

```
public class Second extends First
{
    private int z;
    public Second (int num) {
        super (num);
        this.z = num;
        System.out.println ("Second");
    }
    public int sum() {
        return super.sum() + this.z;
    }
    public void add (First other) {
        this.x += other.getX();
        this.y += other.getY();
        if (other instanceof Second)
            this.z += ((Second)other).z;
        System.out.println("x = "+ this.x +
                           " y = "+ this.y+ " z = " +this.z);
    }
}
```

Обратите внимание: продолжение вопроса на следующей странице.

/продолжение на странице 17/

Дан класс Tester:

```
public class Tester
{
    public static void main(String[] args)
    {
        First f1 = new First (40);
        First f2 = new First (40, 50);
        First f3 = new Second (100);
        Second s1 = new Second (100);
        Second s2 = new Second (100);
        // ***
    }
}
```

(א) Нарисуйте объекты, которые были созданы методом main , и напишите вывод [פלט] этого метода.

(ב) Подставьте каждую из команд 1–10 , приведенных ниже, в метод main , на место, обозначенное выше звездочками: *** .

Напишите в своей тетради номер команды и укажите, является код корректным или некорректным.

Если код корректный, напишите, каков будет вывод [פלט], если код некорректный, объясните почему.

Примечание: нет связи между командами. Иными словами, необходимо рассмотреть каждую команду, как будто она единственная в методе main .

1. System.out.println ("Total = " + First.getCount());
2. System.out.println ("Total = " + Second.getCount());
3. System.out.println ("sum = " + s1.sum());
4. System.out.println ("sum = " + f3.sum());
5. s1=new First (100);
6. f1.add (s2);
7. s1.add (s2);
8. s2.add (f3);
9. ((First)s1).add (f1);
10. s1=new Second (100, 100);

/продолжение на странице 18/

Объектно-ориентированное программирование [תכנות מונחה עצמים] на языке C#

Если вы изучали этот курс и пишите на языке C#, ответьте на один из вопросов 13–14 (25 баллов).

13. В ресторане "На уровне" можно заказать место для одного или более посетителей. В конце трапезы посетители платят по отдельности, каждый за себя. Каждый посетитель может выбрать оплату наличными, или в приложении, или кредитной картой. При оплате кредитной картой есть возможность разделить счет на несколько равных платежей.
Для управления платежной системой для ресторана разрабатывают проект с интерфейсом **IPayment** (Interface) и классами **Cash, App, Credit, Reservation, Restaurant**, которые объясняются ниже.

- **IPayment** – платеж
  в интерфейсе есть метод: double GetPrice()
- **Cash** – оплата наличными
  Атрибуты класса:
  - sumCash – сумма для оплаты наличными, типа действительное число.
  - name – имя посетителя, типа строка.
- **App** – оплата в приложении
  Атрибуты класса:
  - sumApp – сумма для оплаты в приложении, типа действительное число.
  - phoneNumber – номер телефона, типа строка.
- **Credit** – оплата кредитной картой
  Атрибуты класса:
  - num – число платежей, целого типа.
  - part – сумма каждого платежа, типа действительное число.
  - creditNumber – номер кредитной карточки, типа строка.
- **Reservation** – заказ
  Атрибуты класса:
  - date – дата заказа, типа строка.
  - total – суммарный счет всех посетителей вместе в рамках этого заказа, типа действительное число.
  - payments – массив, в котором хранится платеж каждого из посетителей, типа **IPayment** (массив без значений null).
- **Restaurant** – ресторан
  Атрибуты класса:
  - array – массив длиной 10000 , типа **Reservation** .
  - current – количество сделанных заказов, целого типа.

Считайте, что заказы хранятся последовательно в массиве и среди них нет значений null.

Примечание: считайте, что в каждом классе определены методы Get и Set и конструкторы.

Обратите внимание: продолжение вопроса на следующей странице.

/продолжение на странице 19/

(א) **(1)** Начертите схему иерархии, описывающую связь между классами и интерфейсом проекта.

Обозначьте реализацию интерфейса посредством стрелки  ,

а включение – посредством символа  .

(2) Ресторан делает скидки посетителям в соответствии с типом оплаты: посетитель, который платит наличными и сумма платежа которого выше 200 шекелей, имеет право на 10% скидки, посетитель, который платит в приложении, всегда получает 5% скидки, а посетитель, который пользуется кредитной картой, платит полную цену.

Метод интерфейса GetPrice возвращает общий платеж, который платит клиент после учета скидки (если скидки нет, метод возвращает полную цену). Реализуйте метод интерфейса GetPrice в каждом из требуемых классов.

(ב) Напишите в классе **Reservation** метод с заголовком CashTotal. Метод возвращает сумму наличных денег, которую получают при оплате заказа (после учета скидки).

(ג) Ниже приведен метод в классе **Reservation**:

```
public void PrintDetails () {
    for (int i = 0; i< payments.Length; i ++) {
        Console.WriteLine (payments[i].GetDetails());
    }
}
```

Этот метод выводит для каждого посетителя определенные данные по следующим критериям:

– если посетитель заплатил наличными, метод выводит его имя.

– если посетитель заплатил в приложении, метод выводит номер его телефона.

– если посетитель заплатил кредитной картой, метод выводит номер его кредитной карты.

Добавьте в проект методы, необходимые для того, чтобы этот метод выполнял требуемое.

Укажите для каждого метода, который вы добавили, к какому классу или к какому интерфейсу он принадлежит.

Указание: запрещено изменять метод PrintDetails .

/продолжение на странице 20/

14. Даны классы – First и Second:

```
public class First
{
    private static int count = 0;
    protected int x;
    protected int y;
    public First (int num) {
        this.x = num;
        this.y = num;
        count ++;
        Console.WriteLine ("First 1");
    }
    public First (int num1, int num2) {
        this.x = num1;
        this.y = num2;
        count ++;
        Console.WriteLine ("First 2");
    }
    public static int GetCount() {
        return count;
    }
    public int GetX() {
        return x;
    }
    public int GetY() {
        return y;
    }
    public  virtual int Sum() {
        return this.x + this.y;
    }
    public virtual void Add (First other) {
        this.x += other.x;
        this.y += other.y;
        Console.WriteLine ("x = "+this.x +
                                " y = " +this.y);
    }
}
```

```
public class Second : First
{
    private int z;
    public Second (int num) : base (num) {
        this.z = num;
        Console.WriteLine (“Second”);
    }
    public override int Sum() {
        return base.Sum() + this.z;
    }
    public override void Add (First other) {
        this.x += other.GetX();
        this.y += other.GetY();
        if (other is Second)
            this.z += ((Second)other).z;
        Console.WriteLine("x = "+ this.x +
                    " y =" + this.y + " z =" +
this.z);
    }
}
```

Обратите внимание: продолжение вопроса на следующей странице.

/продолжение на странице 21/

Дан класс Tester:

```
public class Tester
{
    public static void Main(string[] args)
    {
        First f1 = new First (40);
        First f2 = new First (40, 50);
        First f3 = new Second (100);
        Second s1 = new Second (100);
        Second s2 = new Second (100);
        // ***
    }
}
```

(א) Нарисуйте объекты, которые были созданы методом Main , и напишите вывод [פלט] этого метода.

(ב) Подставьте каждую из команд 1–10 , приведенных ниже, в метод Main , на место, обозначенное выше звездочками: *** .

Напишите в своей тетради номер команды и укажите, является код корректным или некорректным.

Если код корректный, напишите, каков будет вывод [פלט], если код некорректный, объясните почему.

Примечание: нет связи между командами. Иными словами, необходимо рассмотреть каждую команду, как будто она единственная в методе Main .

1. Console.WriteLine ("Total = " + First.GetCount());
2. Console.WriteLine ("Total = " + Second.GetCount());
3. Console.WriteLine ("sum = " + s1.Sum());
4. Console.WriteLine ("sum = " + f3.Sum());
5. s1 = new First (100);
6. f1.Add (s2);
7. s1.Add (s2);
8. s2.Add (f3);
9. ((First)s1).Add (f1);
10. s1 = new Second (100, 100);

Желаем успеха! **בהצלחה!**