دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف دوم درس آزمون نرم افزار

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: بهار ۱۴۰۱/۱۴۰۲

مدرّس: دکتر الهام محمودزاده

۱

۱.۱ سوال ۶ فصل هفتم صفحه‌ی ۱۷۸

۱.۱.۱ a

- **Node coverage**:

$(1), (2), (3), (4), (5), (6), (7), (8), (9), (10)$

- **Edge coverage**:

$(1,4), (1,5), (2,5), (6,2), (3,6), (3,7), (4,8), (5,8), (5,9), (9,6), (6,10), (7,10)$

- **Prime path coverage**:

$(1,4,8), (1,5,8), (3,6,10), (3,7,10), (1,5,9,6,2), (1,5,9,6,10), (2,5,9,6,10), (2,5,9,6,2), (3,6,2,5,8), (3,6,2,5,9), (5,9,6,2,5), (6,2,5,9,6), (9,6,2,5,8), (9,6,2,5,9)$

۲.۱.۱ b

$(1,4,8), (2,5,9), (3,6,10), (3,7,10)$

۳.۱.۱ c

$(1,4,8), (1,5,9,6,2,5,8), (3,6,10), (3,7,10)$

۲.۱ سوال ۷ فصل هفتم صفحه‌ی ۱۷۹

```
public static int search (List list, Object element)
// Effects: if list or element is null throw NullPointerException
//   else if element is in the list, return an index
//   of element in the list; else return -1
//   for example, search ([3,3,1], 3) = either 0 or 1
//       search ([1,7,5], 2) = -1
```

شکل ۱

Characteristic: Location of element in list
- Block 1: element is first entry in list
- Block 2: element is last entry in list
- Block 3: element is in some position other than first or last

شکل ۲

اگر لیست ما تک عضوی باشد، برای مثال $\{apple\}$ و عنصر مورد جستجوی ما apple باشد، این عنصر در هر دو بلاک می‌تواند قرار گیرد. پس ویژگی "Location of element in list" نمی‌تواند تهی بودن اشتراک بین هر دو بلاک را حفظ کند و در نتیجه فضای ورودی را به درستی افراز کند.

b ۱.۲.۱

اگر عنصر در لیست وجود نداشته باشد هیچ بلاکی این حالت را در نظر نمی‌گیرد. برای مثال $\{apple, orange, banana\}$ و عنصر مورد جستجو $melon$ باشد، هیچ یک از سه بلاک بالا این حالت را پوشش نمی‌دهند.

c ۲.۲.۱

	Block: 1	**Block: 2**
Characteristic: element is the first entry in list	True	False
Characteristic: element is the last entry in list	True	False
Characteristic: element is neither first nor last in the list but is in the list	True	False
Characteristic: element is not in the list	True	False

۳.۱ سوال ۴ فصل ششم

جهت افراز فضای ورودی برای کلاس `GenericStack` می‌توانیم افرازها و بلوک‌های زیر در نظر بگیریم:

1. `GenericStack()` constructor:

- Partition 1: Empty stack creation
- Partition 2: Stack creation with initial capacity
- Partition 3: Stack creation with initial capacity less than 1
- Partition 4: Stack creation with null initial capacity

2. `push(Object X)` method:

- Partition 1: Pushing a non-null object onto an empty stack

- Partition 2: Pushing a non-null object onto a non-empty stack
- Partition 3: Pushing a null object onto an empty stack
- Partition 4: Pushing a null object onto a non-empty stack

3. `pop()` method:

- Partition 1: Popping from an empty stack
- Partition 2: Popping from a non-empty stack

4. `isEmpty()` method:

- Partition 1: Calling `isEmpty()` on an empty stack
- Partition 2: Calling `isEmpty()` on a non-empty stack

این تقسیم‌بندی‌ها فضای ورودی را به سناریوهای خاص تقسیم می‌کنند که جوانب مختلف کلاس `GenericStack` را پوشش می‌دهند. با انتخاب ورودی‌های آزمون از این تقسیم‌بندی‌ها، می‌توانیم پوشش مناسبی از قابلیت‌های ارائه شده توسط این کلاس را به دست آوریم.

1.3.1 a

The input variables for the GenericStack class are:

- **X**: The object to be pushed onto the stack.

The state variables are:

- **Stack**: The data structure used to store the elements in the stack.

1.3.2 b

Characteristics of the input variables:

- **X**:
  - Type: Object
  - Valid values: Any object, including null

1.3.3 c

Characteristics of inputs:

- **Stack size**:
  - Type: Integer
  - Range: Non-negative integers (including zero)
  - Invalid values: Negative values

1.3.4 d

Partitioning the characteristics into blocks:

- Partition 1: **Stack size**
  - Block 1 (*Base*): Zero size
  - Block 2: Non-zero size

1.3.5 e

Values for each block:

- Block 1 (*Stack size*):
  - *Base*: size = 0
- Block 2 (*Stack size*):
  - *Base*: size = 1
  - Additional: size = 2

۲

۱.۲ سوال ۴ فصل ششم صفحه‌ی ۱۳۹

a ۱.۱.۲

بله شرط Completeness را ارضا می‌کند. چون هر ورودی در یکی از سه حالت مذکور دسته‌بندی می‌شود.

b ۲.۱.۲

بله شرط Disjointness را ارضا می‌کند.

c ۳.۱.۲

بله شرط Completeness را ارضا می‌کند.

d ۴.۱.۲

خیر اگر S_1 و S_2 تهی باشند در هر چهار بلاک دسته‌بندی می‌شوند. پس شرط Disjointness را حفظ نمی‌کند.

e ۵.۱.۲

$1 + \sum_{i=1}^{Q} (B_i - 1) = 1 + (3 - 1) + (4 - 1) = 1 + 2 + 3 = 6$

f ۶.۱.۲

Characteristic: relation between s1 and s2

- s1 and s2 represent the same non-empty set
- s1 is a non-empty subset of s2
- s2 is a non-empty subset of s1
- s1 and s2 do not have any elements in common (one or both could be empty)

۲.۲ سوال ۵ فصل ششم صفحه‌ی ۱۳۹

a ۱.۲.۲

$$(Value1, Value2, Operation) \in \{(-1, -1, ' + ') , (0, 0, ' - ') , (1, 1, ' \times ') , (-1, 2, ' \div ')\}$$

b ۲.۲.۲

Base choice: $Value1 \geq 0, Value2 \geq 0, Operation = ' + '$

$$(Value1, Value2, Operation) \in \{(1, 1, ' - ') , (1, 1, ' \times ') , (1, 1, ' \div ') , (1, 0, ' + ') , (1, -1, ' + ') , (-1, 1, ' + ') , (0, 1, ' + ')\}$$

c ۳.۲.۲

$$3 \times 3 \times 4 = 36$$

d ۴.۲.۲

(لطفا زوم کنید :))

$$(Value1, Value2, Operation) \in \{(-1, -1, ' + ') , (-1, 0, ' - ') , (-1, 1, ' \times ') , (-1, -1, ' \div ') , (0, 0, ' + ') , (0, 1, ' - ') , (0, -1, ' \times ') , (0, 0, ' \div ') , (1, 1, ' + ') , (1, -1, ' - ') , (1, 0, ' \times ') , (1, 1, ' \div ')\}$$

۳

(لطفا زوم کنید :))

$$\left(\max_{i=1}^{Q}(B_i)\right) \times \left(\max_{j=1, j \neq i}^{Q}(B_j)\right) = 4 \times 4 = 16 \{(A1, B1, C1, D1), (A1, B2, C2, D2), (A1, B3, C3, D3), (A1, B4, C4, D4), (A2, B1, C2, D3), (A2, B2, C3, D4), (A2, B3, C4, D1), (A2, B4, C1, D2), (A3, B1, C3, D1), (A3, B2, C4, D2), (A3, B3, C1, D3), (A3, B4, C2, D4), (A4, B1, C4, D3), (A4, B2, C1, D4), (A4, B3, C2, D1), (A4, B4, C3, D2)\}$$

۴

4.1 a

The input variables for the BoundedQueue class are:

- **capacity**: The maximum number of elements allowed in the queue.
- **X**: The object to be enqueued.

The state variables are:

- **Queue**: The array or data structure used to store the elements in the queue.
- **Size**: The current number of elements in the queue.
- **Front**: The index of the front element in the queue.
- **Rear**: The index of the rear element in the queue.

4.2 b

Characteristics of the input variables:

- **capacity**:
  - Type: Integer
  - Range: Positive integers (greater than zero)
  - Invalid values: Negative values, zero
- **X**:
  - Type: Object
  - Valid values: Any object, including null

4.3 c

Partitioning the characteristics into blocks:

- Partition 1: **capacity**
  - Block 1 (*Base*): Positive value (e.g., 1)
- Partition 2: **X**
  - Block 2 (*Base*): Non-null object

4.4 d

Values for each block:

- Block 1 (*capacity*):
  - *Base*: capacity = 1
  - Additional: capacity = 2

- Block 2 (*X*):
  - *Base*: X = `new Object()`
  - Additional: X = null