مثال‌هایی از مبحث Fault (نقص)، Error (خطا) و Failure (شکست)

Example 1

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

  for (int i = 0; i < x.length; i++)
  {
    if (x[i] == 0)
    {
      return i;
    }
  }
  return -1;
}
  // test:  x=[0, 1, 0]
  //        Expected = 2
```

(a) Explain what is wrong with the given code. Propose a modification to the code.

(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not.

(c) If possible, give a test case that executes the fault, but does not result in an error state.

If not, briefly explain why not.

(d) If possible give a test case that results in an error state, but not a failure.

Hint: Don't forget about the program counter. If not, briefly explain why not.

(e) For the given test case, describe the first error state. Be sure to describe the complete state

مثال ۱

- a) فالت را شناسایی و سپس ترمیم کنید.

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- **a) فالت را شناسایی و سپس ترمیم کنید.**
- ایندکس اولین وقوع 0 را برمی‌گرداند.
- ترمیم؟
- راه‌حل اول (v1): اجرای معکوس حلقه

For (i=length-1; i>=0; i--)

- راه‌حل دوم (v2):

```
int index = -1; // Added before the loop
index = i; // Replaces return statement inside the loop body
return index; // Replaces return statement after the loop
```

- v1 کارایی بهتری دارد

مثال ۱

- b) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا نکند.

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

مثال ۱

- **b) در صورت امکان، test case ای شناسایی نمایید که fault را اجرا نکند.**
- امکان‌ناپذیر به ازای هر دو روش v1 و v2

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- c) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا کند اما منجر به **error** (وضعیت خطادار) نشود.

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- c) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا کند اما منجر به **error** (وضعیت خطادار) نشود.
- با در نظر گرفتن v1: For (i=length-1; i>=0; i--)
  - آیا برای **x = null** خطا داریم؟
  - آیا برای **x = []** خطا داریم؟
  - آیا برای آرایه تک‌عضوی **x = [d], d ≠ 0** خطا داریم؟
  - آیا برای آرایه تک‌عضوی **x = [0]** خطا داریم؟
  - تنها با مورد تست **x = [0]** خطا اتفاق نمی‌افتد

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- c) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا کند اما منجر به **error** (وضعیت خطادار) نشود.
- با در نظر گرفتن v2:

```
int index = -1; //
index = i; // Repla
return index; // Re
```

- امکان‌ناپذیر
- همواره به ازای هر ورودی، دومین state از کد اصلی و کد صحیح متفاوت خواهد بود:

Second State Original: $x = \ldots$
$i = 0$
$PC =$ *just after* `i= 0;`

Second State Repair: $x = \ldots$
$index = -1$
$PC =$ *just before the for loop*

مثال ۱

- d) در صورت امکان، **test case** ای شناسایی نمایید که منجر به **error** شود اما **failure** بوجود نیاید.

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- d) در صورت امکان، **test case** ای شناسایی نمایید که منجر به **error** شود اما **failure** بوجود نیاید.
- با در نظر گرفتن هر دو روش v1 و v2:
- **x = null**
- **x = []**
- **هر آرایه‌ای که که حداکثر یک عضو صفر دارد**

  (به استثنای x = [0] برای v1)

مثال ۱

- e) برای test case داده شده در مسئله، اولین وضعیت دارای error (خطا) را مشخص نمایید.

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

مثال ۱

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
     if (x[i] == 0)
     {
       return i;
     }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

- **e) برای test case داده شده در مسئله، اولین وضعیت دارای error (خطا) را مشخص نمایید.**
- با در نظر گرفتن v1:
- First Error State:
- (x= [0, 1, 0], i=0, PC= *just after i= 0 statement*)
- با در نظر گرفتن v2:
- اولین state خطادار درست پس از محل اجرای کد فراموش‌شده (یعنی int index = -1) اتفاق می‌افتد.

Example 2

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
  int count = 0;
  for (int i = 0; i < x.length; i++)
  {
    if (x[i]% 2 == 1 || x[i] > 0)
    {
      count++;
    }
  }
  return count;
}
  // test:  x=[-3, -2, 0, 1, 4]
  //        Expected = 3
```

(a) Explain what is wrong with the given code. Propose a modification to the code.

(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not.

(c) If possible, give a test case that executes the fault, but does not result in an error state.

If not, briefly explain why not.

(d) If possible give a test case that results in an error state, but not a failure.

Hint: Don't forget about the program counter. If not, briefly explain why not.

(e) For the given test case, describe the first error state. Be sure to describe the complete state

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- a) فالت را شناسایی و سپس ترمیم کنید.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- **a) فالت را شناسایی و سپس ترمیم کنید.**
- برنامه نمی‌تواند اعداد فرد منفی را بشمارد.
- با استفاده از شرط if (x[i] % 2 == 1)، برنامه نمی‌تواند اعداد فرد منفی که باقیمانده تقسیم آنها بر ۲ برابر با 1- می‌شود را در شمارش لحاظ کند.
- ترمیم؟
- if (x[i]% 2 == -1 || x[i] > 0)
- تمامی اعداد مثبت با استفاده از شرط دوم شمارش می‌شوند و اعداد منفی فرد هم با استفاده از شرط اول شمارش می‌شوند.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- b) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا نکند.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- **b) در صورت امکان، test case ای شناسایی نمایید که fault را اجرا نکند.**
- x=[]
- اگر آرایه مورد نظر خالی باشد، در آنصورت شرط ورود به حلقه برقرار نمی‌شود و در نتیجه داخل حلقه که محل وقوع fault است، اجرا نمی‌شود. در این حالت مقدار count=0 بعنوان خروجی برگردانده می‌شود.
- X= null
- با اجرای دستور i<x.lenght، برنامه NullPointerException برمی‌گرداند و در نتیجه بدنه حلقه اجرا نمی‌شود.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- c) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا کند اما منجر به **error** (وضعیت خطادار) نشود.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- c) در صورت امکان، **test case** ای شناسایی نمایید که **fault** را اجرا کند اما منجر به **error** (وضعیت خطادار) نشود.
- آرایه ای غیر خالی که فاقد اعداد فرد منفی باشد
  - مثلا: x=[1, 3, 8]
- زمانی که تمام اعداد آرایه مثبت یا منفی زوج باشند، با وجود اینکه fault اجرا می‌شود ولی منجر به خراب شدن وضعیت داخلی برنامه نمی‌شود. دلیل آن این است که برای اعداد مثبت یا منفی زوج، اجرای شرط (if (x[i] % 2 == 1 عملا تاثیری نخواهد داشت. در این حالت اعداد مثبت با توجه به شرط دوم شمارش می‌شوند و اعداد منفی زوج هم که در هر صورت نباید شمارش شوند.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- d) در صورت امکان، **test case** ای شناسایی نمایید که منجر به **error** شود اما **failure** بوجود نیاید.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- **d) در صورت امکان، test case ای شناسایی نمایید که منجر به error شود اما failure بوجود نیاید.**
- امکان‌ناپذیر
- در این برنامه هرگاه error اتفاق بیفتد، منجر به failure خواهد شد.
- دلیل آن این است که error زمانی اتفاق می‌افتد که آرایه حاوی حداقل یک عدد منفی فرد باشد. در این صورت در حین اجرای برنامه هرگاه به اولین عدد منفی فرد برسیم مقدار count نادرست می‌شود (وضعیت داخلی نادرست) و درنتیجه تمامی وضعیتهای بعدی در حین اجرا دارای error خواهند بود. در نهایت نیز پس از خاتمه برنامه، مقدار نهایی count اشتباه است و سیستم با failure روبرو می‌شود.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //          Expected = 3
```

- e) برای **test case** داده شده در مسئله، اولین وضعیت دارای **error** (خطا) را مشخص نمایید.

مثال ۲

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]% 2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

- e) برای **test case** داده شده در مسئله، اولین وضعیت دارای **error** (خطا) را مشخص نمایید.
- اولین وضعیت خطا زمانی رخ می‌دهد که در خانه اول آرایه اجرای شرط if منجر می‌شود که عدد 3- که یک عدد فرد است، به اشتباه شمارش نشود و درنتیجه مقدار count به اشتباه اضافه نشود و 0 باقی بماند:

- First Error State:
- (x= [-3, -2, 0, 1, 4], i=0, count=0, PC= *at end of* **if** *statement*)