Ex 0bb “inventory” – Test black box

A retail support system manages an inventory of items. Each item has a descriptor and the number of available items.

```
/*  Inventory */
typedef struct item_tag {
  char* itemCode; // unique identifier, a.k.a. bar code
  int availability; // number of items avai lable
  char* description; // description of item
  char* name;  // name of item
} item
#define NON_EXISTENT_ERROR -1
#define DUPLICATE_ERROR -2
#define NOT_AVAILABLE_ERROR -3

// creates a new item
item* new_item(char* code, int quantity);

// adds new descriptor
// (returns 0 if ok, DUPLICATE_ERROR if item already exists)
int addItem(item* i);

// returns item with given code
item* searchItem (char* itemCode);

// returns available items (NON_EXISTENT_ERROR is item does not exist)
int availabilityItem(char* itemCode);

// subtracts 1 to availability
// (returns NON_EXISTENT_ERROR is item does not exist,
//          NOT_AVAILABLE_ERROR if not available)
int subtractItem (char* itemCode);

//add given quantity to item availability
// (returns NON_EXISTENT_ERROR if item does not exist)
int addQtyToItem(String itemCode, int qty_to_add);

// subtract given quantity to item availability
// (returns NON_EXISTENT_ERROR is item does not exist,
//          NOT_AVAILABLE_ERROR if not available)
int  subtractQtyToItem(String itemCode, int qty_to_add);
```

1. Define equivalence classes, and related boundary condition
2. For each class, define at least one test case

Ex 1bb "Events"– Test black box

A queue of events in a simulation system receives events. Each event has a time tag.
It is possible to extract events from the queue; the extraction must return the event with lowest time tag.
The queue discards events with negative or null time tag.
The queue must accept at most 100.000 events.
Events with the same time tag must be merged (i.e. the second received is discarded)

```
#define OK 0
#define INVALID_TAG -1
#define QUEUE_OVERFLOW -2
#define QUEUE_EMPTY -3
#define QUEUE_SIZE 100000

/**
 * cancels all events
 */
void reset();


/**
 * pushes a time tag into the queue.
 * Discards events with time tag already existing (OK)
 * and with negative or zero time tag (INVALID_TAG)
 * checks for overflow (QUEUE_OVERFLOW)
 */
int push(int timeTag);

/**
 * returns and cancels the event with lowest time tag.
 * Returns QUEUE_EMPTY on empty queue
 */
int pop();
```

1. Define equivalence classes, and related boundary condition
2. For each class, define at least one test case

Ex 2bb “int converter”– Test black box

A function converts a sequence of chars in an integer number. The sequence can start with a ‘-‘ (negative number). If the sequence is shorter than 6 chars, it is filled with blanks (to the left side).
The integer number must be in the range minint = -32768 to maxint = 32767.
The function signals an error if the sequence of chars is not allowed.
The sequence of chars must be <= 6 chars

1 define equivalence classes, and related boundary condition
2 for each class, define at least one test case
3 define class Cartesian product of classes

```
int convert(string s)
```

“1” → “ 1” → 1
“1234” → 1234
“1234567” → error
“99999” → error
“12ff99” → error

.
keeping classes separate

Ex 3bb "calendar" – Test black box

Consider a method that returns the number of days in a month, given the month and year.

```
int getNumDaysInMonth(int m, int y);
```

The month and the year are specified as integers. By convention, 1 represents the month of January 2 the month of February, and so on. The range of valid inputs for the year is 0 to maxInt.

1 define equivalence classes, and related boundary condition
2 for each class, define at least one test case

Ex 4bb “sort integers” – Test black box

The program shall take as input an array of three integer numbers. The program shall output the greatest number among the elements of the array. The program shall order the elements of the array in decreasing order.

```
int maxSort(int[] a);
```

1. Identify the equivalence classes for the input

Ex 5bb "parallelogram" – Test black box

The function

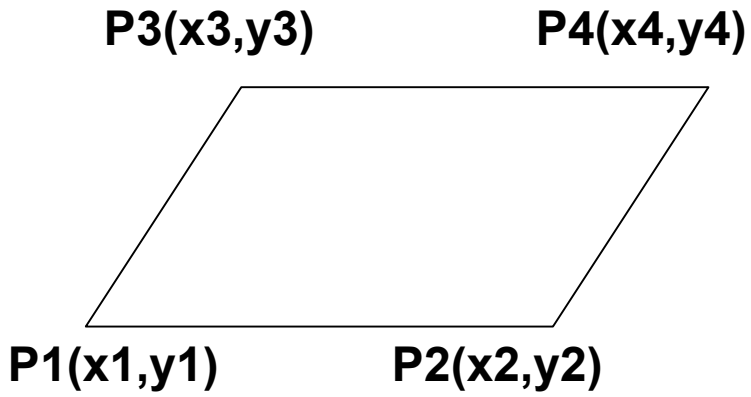
```
parallelogram(int x1, int x2, int x3, int x4,
              int y1, int y2, int y3, int y4)
```

computes the area of a parallelogram.

Requirements are:

- area is always strictly > 0;
- the parallelogram should stay in the first quadrant of the Cartesian plan;
- coordinates have the following meaning:

In case of error or invalid input, -1 is returned.



1. define the equivalence classes, and related boundary condition
2. for each class, define at least one test case

Ex 6bb "router" – Test black box

The big company "P2" has a private network in which center an internal router was placed. The router accepts packets of variable but limited length (max 1500 Bytes), and before transmitting them in the output interfaces, it performs some conformance checks:

- no more than one packet every 0.1 ms;
- destination IP must belong to the subnet 192.168.1.0;
- if source IP is 192.168.1.1, any destination is reachable;

Buffer is 1,5MB wide, packet process time is 0.2 ms .

In order to test the router, a software driver was written. The main function is:

```
int sendPacket(String srcIP, String destIp,
                  int bytes, float time_ms);
```

The function returns the current buffer occupancy (Bytes) if packet is accepted, -1 otherwise.

1 define equivalence classes, and related boundary condition
2 for each class, define at least one test case