Experiment 2 – Fixed , Dynamic Arrays and Queues

Part 1:

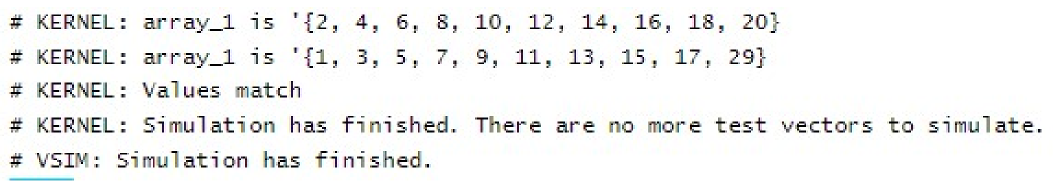
Aim:

1. Write the System Verilog code to:
   i. Declare two bit-type fixed-size array of size 10 which can store 8bits values. Initialize first array with values {2,4,6,8,10,12,14,16,18,20} and second array with values {1,3,5,7,9,11,13,15,17,29}.
   ii. Display the values stored in two array declared in step-i.
   iii. Copy the contents of first array into second array and compare two arrays and display the result of comparison.

Code:

```
module arrays();
  bit [7:0] array_1[10], array_2[10];
  int x;

  initial begin
    array_1 = '{2,4,6,8,10,12,14,16,18,20};
    array_2 = '{1,3,5,7,9,11,13,15,17,29};

    $display("array_1 is ",array_1);
    $display("array_1 is ",array_2);

    array_2 = array_1;
    foreach(array_1[i]) if(array_1[i] != array_2[i]) x = x+1;

    if(x) $display("Values dont match");
    else $display("Values match");

  end
endmodule
```

Output:



```
# KERNEL: array_1 is '{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
# KERNEL: array_1 is '{1, 3, 5, 7, 9, 11, 13, 15, 17, 29}
# KERNEL: Values match
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Part 2:

Aim:

Write the SystemVerilog code to:

i. Declare the two integer type dynamic array.

ii. Store the following values in the first dynamic array-{ 3,6,9,12,15,18} and {2,4,6,8,10,12,14} in the second dynamic array.

iii. Print the sum of the elements stored in the arrays.

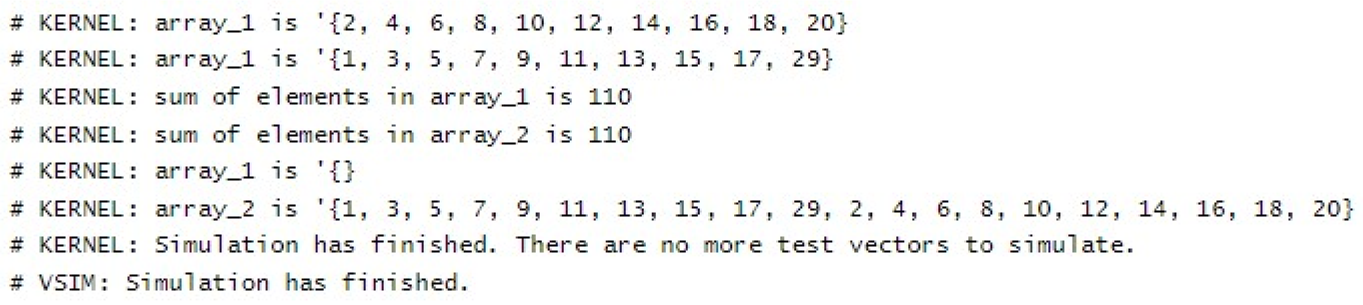
iv. Insert the contents of the first array into second array (after its last element).

v. Delete the first array and try to print the contents of the first and second array. Write remark if you have any observation.

Code:

```
module arrays();
  integer array_1[], array_2[];
  int x,y;

  initial begin
    array_1 = new[10];
    array_1 = {2,4,6,8,10,12,14,16,18,20};
    array_2 = new[10];
    array_2 = {1,3,5,7,9,11,13,15,17,29};

    $display("array_1 is ",array_1);
    $display("array_1 is ",array_2);

    foreach(array_1[i]) x = x+array_1[i];
    $display("sum of elements in array_1 is %0d",x);
    foreach(array_2[i]) y = y+array_2[i];
    $display("sum of elements in array_2 is %0d",y);

    array_2 = new[20]({array_2,array_1});
    array_1.delete();

    $display("array_1 is ",array_1);
    $display("array_2 is ",array_2);
  end
endmodule
```

Output:



```
# KERNEL: array_1 is '{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
# KERNEL: array_1 is '{1, 3, 5, 7, 9, 11, 13, 15, 17, 29}
# KERNEL: sum of elements in array_1 is 110
# KERNEL: sum of elements in array_2 is 110
# KERNEL: array_1 is '{}
# KERNEL: array_2 is '{1, 3, 5, 7, 9, 11, 13, 15, 17, 29, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Part 3:

Aim:

Write the SystemVerilog code to:

i. Declare the two Queues Queue_1 and Queue_2. Initialize the first queue with values {3,4,5,6} and second queue with values {10,11,12,13}.

ii. Insert 10 before 4 in queue_1 and display the contents.

iii. Insert Queue_2 in Queue_1 after the index value 3. Display the contents of the Queue_1.

iv. Delete value 3 from the first queue. Display the contents of the Queue_1.

v. Insert value 20 at the front of Queue_1. Display the contents of the Queue_1.

vi. Pop the last element of the Queue _1. Display the contents of the Queue_1.

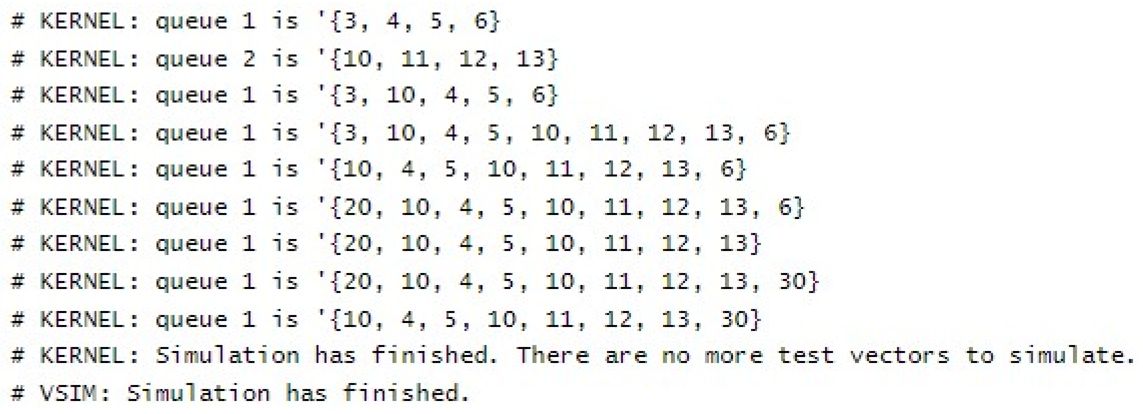
vii. Insert value 30 at the back of Queue_1. Display the contents of the Queue_1.

viii. Pop the front element of the Queue _1. Display the contents of the Queue_1.

Code:

```
module arrays();
 int x;
  int queue_1[$],queue_2[$];
  initial begin
    int queue_1[$] = {3,4,5,6};
    int queue_2[$] = {10,11,12,13};

    $display("queue 1 is ",queue_1);
    $display("queue 2 is ",queue_2);

    queue_1.insert(1,10);
    $display("queue 1 is ",queue_1);

    foreach(queue_2[i]) queue_1.insert(4+i,queue_2[i]);
    $display("queue 1 is ",queue_1);
    queue_1.delete(0);
    $display("queue 1 is ",queue_1);

    queue_1.push_front(20);
    $display("queue 1 is ",queue_1);

    x = queue_1.pop_back();
    $display("queue 1 is ",queue_1);

    queue_1.push_back(30);
    $display("queue 1 is ",queue_1);

    x = queue_1.pop_front();
    $display("queue 1 is ",queue_1);
  end
endmodule
```

Output:



```
# KERNEL: queue 1 is '{3, 4, 5, 6}
# KERNEL: queue 2 is '{10, 11, 12, 13}
# KERNEL: queue 1 is '{3, 10, 4, 5, 6}
# KERNEL: queue 1 is '{3, 10, 4, 5, 10, 11, 12, 13, 6}
# KERNEL: queue 1 is '{10, 4, 5, 10, 11, 12, 13, 6}
# KERNEL: queue 1 is '{20, 10, 4, 5, 10, 11, 12, 13, 6}
# KERNEL: queue 1 is '{20, 10, 4, 5, 10, 11, 12, 13}
# KERNEL: queue 1 is '{20, 10, 4, 5, 10, 11, 12, 13, 30}
# KERNEL: queue 1 is '{10, 4, 5, 10, 11, 12, 13, 30}
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Result:

The given problem statement is executed and verified to be correct.