	D /				1 4 757
	Bước	call stack	macrotask queue	microtask queue	web APIs
	<u> </u>	1 clearOutput()			
Sync		2 output("BEGIN sync")			
		3 sFunction();			
		4 output(ret="Hello S!");			
		5 output("END.")			
	Bước	call stack	macrotask queue	microtask queue	web APIs
		1 clearOutput()			
		2 output("BEGIN settimeout");			
		setTimeout(function() { let ret =			
		3 sFunction(); output(ret); }, 0);			
Timeout		4			function()
		5	sFunction();		()
			output(ret);		
	h	6 output("END.");			
		7 sFunction():	output(ret);		
		8 output(ret="Hello S!");			
		o output(ret= reno b.),		1	l
	Bước	call stack	macrotask queue	microtask queue	I
		1 clearOutput()			
		2 output("BEGIN promise")			Ī
		aFunction1().then(function (ret)	{		
Promise		3 output(ret); });			
				new Promise(function(fulfill, reject) {	Ī
		4 output("END.")	output(ret);	fulfill("Hello A1!"); })	
		5 fulfill("Hello A1!"); //ret="Hello	A1!" output(ret);		Ī
		6 output(ret);			
			T		т
	Bước	call stack	macrotask queue	microtask queue	
		1 clearOutput()			
		2 output("BEGIN async")			
		aFunction2().then(function (ret)	{		
Async	<u> </u>	3 output(ret); });			1
				new Promise(function(fulfill, reject) {	
		4 output("END.")	output(ret);	fulfill("Hello A2!"); })	
		5 fulfill("Hello A2!"); //ret="Hello	A2!" output(ret);		
		6 output(ret);			
	Bước	call stack	macrotask queue	microtask queue	Ī
	Duoc	1 clearOutput()	macrotask queue	inicrotask queue	ł
	-	2 output("BEGIN await")			+
		3 aFunction1();			ł
Await		5 aruncuon 1();		new Promise(function(fulfill, reject) {	-
Await		4 //chờ aFunction() trả về xong kết	aná	fulfill("Hello A1!"); })	
		5 fulfill("Hello A1!"); //ret="Hello		iumii(rieno Ar:); })	+
	-	6 output(ret);	AI:		1
		7 output("END.");			+
	J	/ Jourpull END.);	1		l

Async vs	Bước	call stack	macrotask queue	microtask queue
	1	clearOutput()		
	2	output("BEGIN sync vs async")		
		aFunction1().then(function (ret) {		
	3	output(ret); });		
		aFunction2().then(function (ret) {		new Promise(function(fulfill, reject) {
	4	output(ret); });	output(ret);	fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject)
Sylic	5		output(ret);	fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject)
			output(ret);	fulfill("Hello A2!"); })
				new Promise(function(fulfill, reject)
	6	sFunction();	output(ret);	fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject)
			output(ret);	fulfill("Hello A2!"); })
				new Promise(function(fulfill, reject)
	7	output(ret="Hello S!");	output(ret);	fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject)
			output(ret);	fulfill("Hello A2!"); })
				new Promise(function(fulfill, reject)
	8	output("END.")	output(ret);	fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject)
			output(ret);	fulfill("Hello A2!"); })
	9			new Promise(function(fulfill, reject)
		fulfill("Hello A1!"); //ret="Hello A1!"	output(ret);	fulfill("Hello A2!"); })
			output(ret);	
				new Promise(function(fulfill, reject)
	10	output(ret);	output(ret);	fulfill("Hello A2!"); })
	11	fulfill("Hello A2!"); //ret="Hello A2!"	output(ret);	
	12	output(ret);		

	Bước	call stack	macrotask queue	microtask queue	
	1	clearOutput()			
	2	output("BEGIN waitall");			
	3	aFunction1();			
				new Promise(function(fulfill, reject) {	
	4	aFunction2();		fulfill("Hello A1!"); })	
	5	Promise.all([p1, p2]).then(function (arr) { output(arr[0]); output(arr[1]); });		new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })	
Wait all				new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })	
			output(arr[0]);	new Promise(function(fulfill, reject) {	
wait an	6	output("END.");	output(arr[1]);	fulfill("Hello A1!"); })	
				new Promise(function(fulfill, reject) {	
				fulfill("Hello A2!"); })	
				Promise.all([p1, p2])	
			output(arr[0]);	new Promise(function(fulfill, reject) {	
	7	fulfill("Hello A1!");	output(arr[1]);	fulfill("Hello A2!"); })	
				Promise.all([p1, p2])	
		fulfill("Hello A2!"); //arr=["Hello	output(arr[0]);		
	8	A1!", "Hello A2!"];	output(arr[1]);	Promise.all([p1, p2])	
	//Promise.all() hoàn thành do cá p1 và p2 đã thành công				
	9	output(arr[0]); output(arr[1]);			

		T	Ť	T	1
	Bước 1	call stack	macrotask queue	microtask queue	web APIs
Mixed	2	clearOutput() output("BEGIN mixed");			
		setTimeout(() => { output('setTimeout			
	3	1'); }, 10);			0-2 (
	4	setTimeout(() => { output('setTimeout 2'); }, 0);			() => { output('setTimeout 1'); }, 10
	5		output(res); return "Sub Promise 1";}	new Promise() //promise 1	() => { output('setTimeout 1'); }, 10
					() => { output('setTimeout 2'); }, 0
	6		output(res); return "Sub Promise 1";}	new Promise() //promise 1	() => { output('setTimeout 1'); }, 10 () => {
			output(res); return "Sub Promise 2";}	new Promise() //promise 2	output('setTimeout 2'); }, 0
			output(res);		
	7		//ret="Promise 1"; return "Sub Promise		() => { output('setTimeout 1');
		fulfill('Promise 1');	1";}	new Promise() //promise 2	}, 10
			output(res); return "Sub Promise 2";}		() => { output('setTimeout 2'); }, 0
			output(res);		j, o
			//ret="Promise 1";		() => {
	8	fulfill('Promise 2');	return "Sub Promise 1";}		output('setTimeout 1'); }, 10
		7	output(res);		
			//ret="Promise 2"; return "Sub Promise		<pre>() => { output('setTimeout 2');</pre>
			2";}		}, 0
	9	output(res="Promise 1");			() => {
			return "Sub Promise 1";	new Promise() //sub promise 1	output('setTimeout 1'); }, 10
			output(res);		
			//ret="Promise 2"; return "Sub Promise		() => { output('setTimeout 2');
			2";}		}, 0
			output(res);		0 . 1
	10	output(res="Promise 2");	return "Sub Promise 2";		() => { output('setTimeout 1');
				new Promise() //sub promise 1	}, 10
			output(res); output(res);	new Promise() //sub promise 2	() => { output('setTimeout 2');
					}, 0
					0 => {
	11	fulfill(Sub Promise 1');	output(res); //ret="Sub Promise 1"	new Promise() //sub promise 2	output('setTimeout 1');
					}, 10
					() => { output('setTimeout 2');
			output(res);		}, 0
			output(res); //ret="Sub		() => { output('setTimeout 1');
	12		Promise 1"		}, 10
			output(ros): //rot_"C_1		() => {
			output(res); //ret="Sub Promise 2"		output('setTimeout 2'); }, 0
	13	output(res="Sub Promise 1");	output(res); //ret="Sub Promise 2"		() => {
					output('setTimeout 1'); }, 10
					() => {
					output('setTimeout 2'); }, 0
	14	output(res="Sub Promise 2");			() => {
					output('setTimeout 1');
					}, 10 () => {
					output('setTimeout 2'); }, 0
					() => { output('setTimeout 1');
	15		output('setTimeout 2');		}, 10
		output('setTimeout 2');	output('setTimeout 1');		
	17	output('setTimeout 1');			