Concurrent Programming Coursework

Summary: You are required to implement the parallel *prefix scan* algorithm in c/mpi:

Specification:

- To have keyboard input of the number, n, of elements in the sequence
- > Process 0 will generate n random integers and broadcast these to the other processes as appropriate
- > Process 0 prints the input and output sequences onto the screen.

Note: use of MPI Scan is explicitly forbidden.

Methodology:

There will be an in-class discussion of the algorithm in which we shall work together to identify the key issues and details of the algorithm – be prepared to ask questions in that session.

Deliverables:

1) A single sor Signement and the toppic of the will contain all necessary c/mpi code. Codes provided for other frameworks (e.g. implemented using Microsoft Visual

11 be awarded a mark of 0)

Your code must https://eduassistpro.github.io/umamiliar with the prefix scan algorithm to understand and be a it.

- Good quality in cold om hyner (is lagated edu_assist_pro
- Explicit testing showing sets of inputs and expected outputs is required
- 2) A very brief pdf report highlighting any significant code features to which you wish to draw attention and the results of your testing to convince the client of correct functionality and any comments on the limitations and scope for upgrade of the code.

Submission is only accepted via the assessment link on the Moodle page.

Marks and feedback will be provided within 15 working days of the submission deadline and each student will receive their completed copy of the marking rubric posted on Moodle. Further feedback can subsequently be requested by email.

Plagiarism: Students will naturally discuss the assignment between themselves, however each student's submission must be wholly their own work. It is easier than you might think to spot both students sharing code and the use of third-party code downloaded from the internet etc. The full force of the University's academic misconduct regulations will be invoked for any infringements.

The Prefix Scan Algorithm

Definition: Given a sequence, \boldsymbol{a} , of input values $(a_0, a_1, a_2, a_3, a_4 \dots a_{N-1})$

the output sequence, **b**, is defined as $(b_0, b_1, b_2, b_3, b_4 \dots b_{N-1})$ where $b_i = \sum_{i=0}^i a_i$

We note that $b_i = b_{i-1} + a_i$

Example: a = (1, 2, 3, 4, 5, 6, 7, 8) then b = (1, 3, 6, 10, 15, 21, 28, 36)

A simple serial C implementation of this algorithm might be

```
void prefix\_scan(int\ n,\ int\ a[],\ int\ b[]){
b[0]=a[0];
for(int\ i=1;i< n;++i)\ b[i]=b[i-1]+a[i];}
```

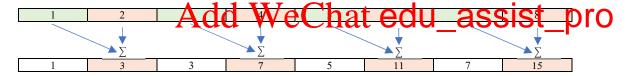
Parallel Prefix Scan Algorithm

Initialise b with a

Assignment Project Exam Help

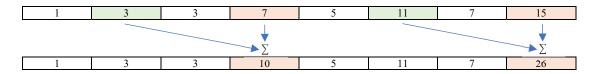
Up Phase

Note those elements in cells Lach such cell's value is repleted the same value. The cells in cells have before it. All other cells keep the same value.



Note those elements in cells of index 4, 8, 12, 16 ..., i.e. spaced by steps of 4

Each such cell's value is replaced by the sum of the current value and the value in the cell 2 places before it. All other cells keep the same value



Note those elements in cells of index 4, 8, 12, 16 ..., i.e. spaced by steps of 8

Each such cell's value is replaced by the sum of the current value and the value in the cell 4 places before it. All other cells keep the same value

1	3	3	10	5	11	7	26
1	3	3	10	5	11	7	36

This is continued until, as just reached in this example, there are no further changes.

Down Phase

Note those elements in cells of index 4, 8, 12, 16 ... i.e. spaced by steps of 4 (i.e. one less than the highest reached in the up phase)

Each such cell's value contributes to a sum of the current value and the value in the cell 2 places after it. All other cells keep the same value

1	3	3	10	5	11	7	26					
\sum												
1	3	3	10	5	21	7	36					

Note those elements in cells of index 2, 4, 6,8 ...,i.e. spaced by steps of 2

Each such cell's value is contributes to a sum of the current value and the value in the cell 1 places after it. All other cells keep the same value

	1	3	3	10	5	21	7	26
_	-		1	10		21	,	20
			▼		\downarrow			
			Σ		Σ		Σ	
Γ	1	3	6	10	15	21	28	36

We could compress this into the following to succinctly show the evolution to the final result.

	Δος	ionr	nent	Proje	act F	vam	Helr	1
1	1 ₂ 101)1 3 111.	HEIIT	1 150		AGIII	118/1	Noting cells spaced by steps of 2
1	3	3	7	5	11	7	15	Noting cells spaced by steps of 4
1	3						26	Noting cells spaced by steps of 8
1	3		,,				36	. ,
		<u>httr</u>	s://e	диас	ecietr	aro a	<u>ithur</u>	$\Omega \setminus \Omega \setminus \Omega$
1	3	1111	10.//C	uuu		71 0. 9	111 36UK	Noting cells spaced by steps of 8
1	3	3	10	5	11	7	36	Noting cells spaced by steps of 4
1	3	3	10	5			36	Noting cells spaced by steps of 2
1	3	& .1	1 10 /	CIE.	t adu	-006	31	ro
		\mathbf{A}^{\prime}		•) ()

Outlined dells are the noted ones per round and shaded cells are thos

A different example illustrates the need to pad with zeros the total number of elements to a power of 2

3	7	1	11	8	20	4	-1	4	0	0	0	0	0	0	0
3	10	1	12	8	28	4	3	4	4	0	0	0	0	0	0
3	10	1	22	8	28	4	31	4	4	0	4	0	0	0	0
3	10	1	22	8	28	4	53	4	4	0	4	0	0	0	4
3	10	1	22	8	28	4	53	4	4	0	4	0	0	0	57
3	10	1	22	8	28	4	53	4	4	0	4	0	0	0	57
3	10	1	22	8	28	4	53	4	4	0	4	0	0	0	57
3	10	1	22	8	28	4	53	4	4	0	57	0	0	0	57
3	10	1	22	8	50	4	53	4	57	0	57	0	57	0	57
3	10	11	22	30	50	54	53	57	57	57	57	57	57	57	57