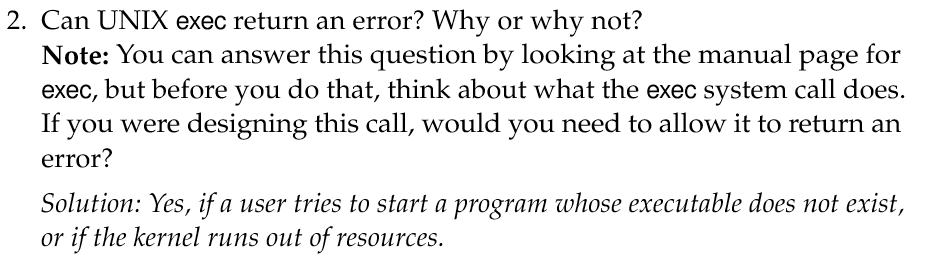
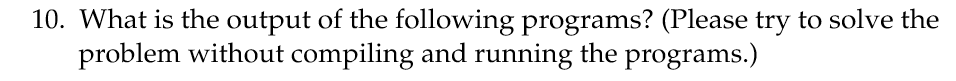
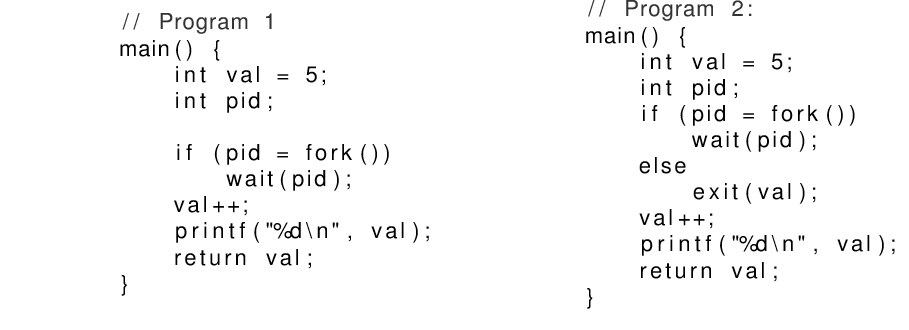
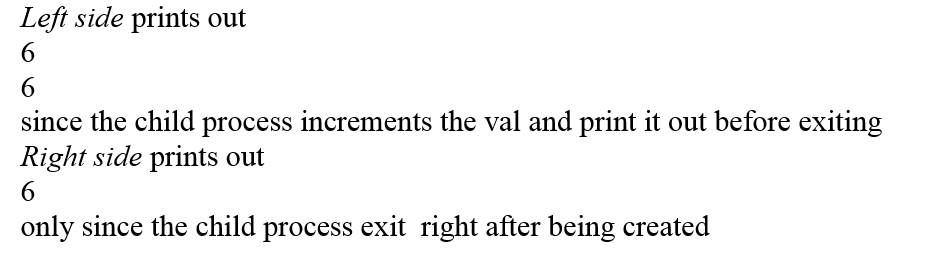
**Chapter 3**



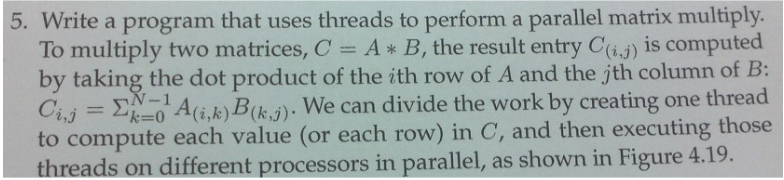




*Solution*:



**Chapter 4**



#include <pthread.h>

#include <stdio.h>

#include <math.h>

const int ROW, COLA, COL, TOTAL\_THREADS = ROW \* COL;

pthread\_t tid[TOTAL\_CUBE];

int a[ROW][COLA], b[COLA][COL], c[ROW][COL];

typedef struct node {

int i;

int j;

} Indices;

void \*matrix(void \*);

int main() {

// should initialize matrixes A and B first

for (int i = 0; i < ROW; i++)

for (int j = 0; j < COL; j++)

{

Indices inices;

indices.i = i;

indices.j = j;

pthread\_create(&tid[i\*COL+j], NULL, matrix, &indices);

}

for (int i = 0; i < TOTAL\_THREADS; i++)

pthread\_join(tid[i], NULL);

for (int i = 0; i < ROW; i++)

for (int j = 0; j < COL; j++)

printf("c[%d][%d] = %d\n", i, j, c[i][j]);

return 0;

}

void \*matrix(void \*indices) {

Indices \*p = (Indices \*)indices;

c[p->i][p->j] = 0;

for(int k=0;k<COLA;k++)

c[p->i][p->j] += a[p->i][k] \* b[k][p->j];

pthread\_exit(NULL);

}