```c
#include "apue.h"
#include <pthread.h>

void *SommaP(void*); /*operazione che effettua il thread*/

struct{
    pthread_mutex_t mutex;
    int minimo;
}shared = {PTHREAD_MUTEX_INITIALIZER, 65535};

/* matrice condivisa tra i thread*/
int **mat;

/*numero righe colonne e righe informazione che devono sapere tutti i
  thread*/
int n;

int main(int argc,char**argv)
{
    int i,j;
    pthread_t *t;
    /*se non viene inserito un dato in ingresso esce*/

    if(argc!=2){
    printf("errore dati in ingresso");
    exit(1);
    }

    n=atoi(argv[1]);
    int *tids[n];

    /*allocazione dinamica matrice*/
    mat=(int**)malloc(n*sizeof(int*));

    for (i=0;i<n;i++)
        mat[i] = (int*)malloc(n*sizeof(int));

    /*inserimento dei dati nella matrice casualmente*/
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
        mat[i][j]=rand()%256;
        printf("%d\t",mat[i][j]);
    }

    printf("\n");
    }

    /*creazione dinamica del vettore di thread*/
    t=(pthread_t*)malloc(n*sizeof(pthread_t));
```

```
49      for(i=0;i<n;i++){
50          tids[i]=(int*)malloc(sizeof(int));
51      *tids[i]=i;
52          pthread_create(&t[i],NULL,SommaP,(void*)tids[i]);
53      }
54
55      /*attesa fine thread*/
56    for(i=0;i<n;i++){
57      pthread_join(t[i],NULL);
58    }
59
60      /*stampa minimo delle somme parziali*/
61      printf("il minimo delle somme parziali e' %d\n",shared.minimo);
62      exit(0);
63  }
64
65  void *SommaP(void* in){
66      int riga,j,somma=0;
67      /*riga su quale effettuare somma parziale*/
68
69      riga=*((int*)in);
70
71      /*se il valore e' pari somma le porzioni pari altrimenti le dispari*/
72      if ((riga%2)==0)
73      for(j=0;j<n;j=j+2) somma=somma+mat[riga][j];
74      else
75      for(j=1;j<n;j=j+2) somma=somma+mat[riga][j];
76
77    /*stampa la sua somma parziale*/
78    printf("sono thread %d, somma parziale %d\n",riga,somma);
79
80      /*accesso esclusivo sulla variabile minimo*/
81    pthread_mutex_lock(&shared.mutex);
82
83      /*verifica se la somma e minore del minimo*/
84      if(somma < shared.minimo)
85          shared.minimo = somma;
86
87      /*rilascia il semaforo*/
88    pthread_mutex_unlock(&shared.mutex);
89
90      /*fine thread*/
91    pthread_exit(0);
92  }
93
94
```