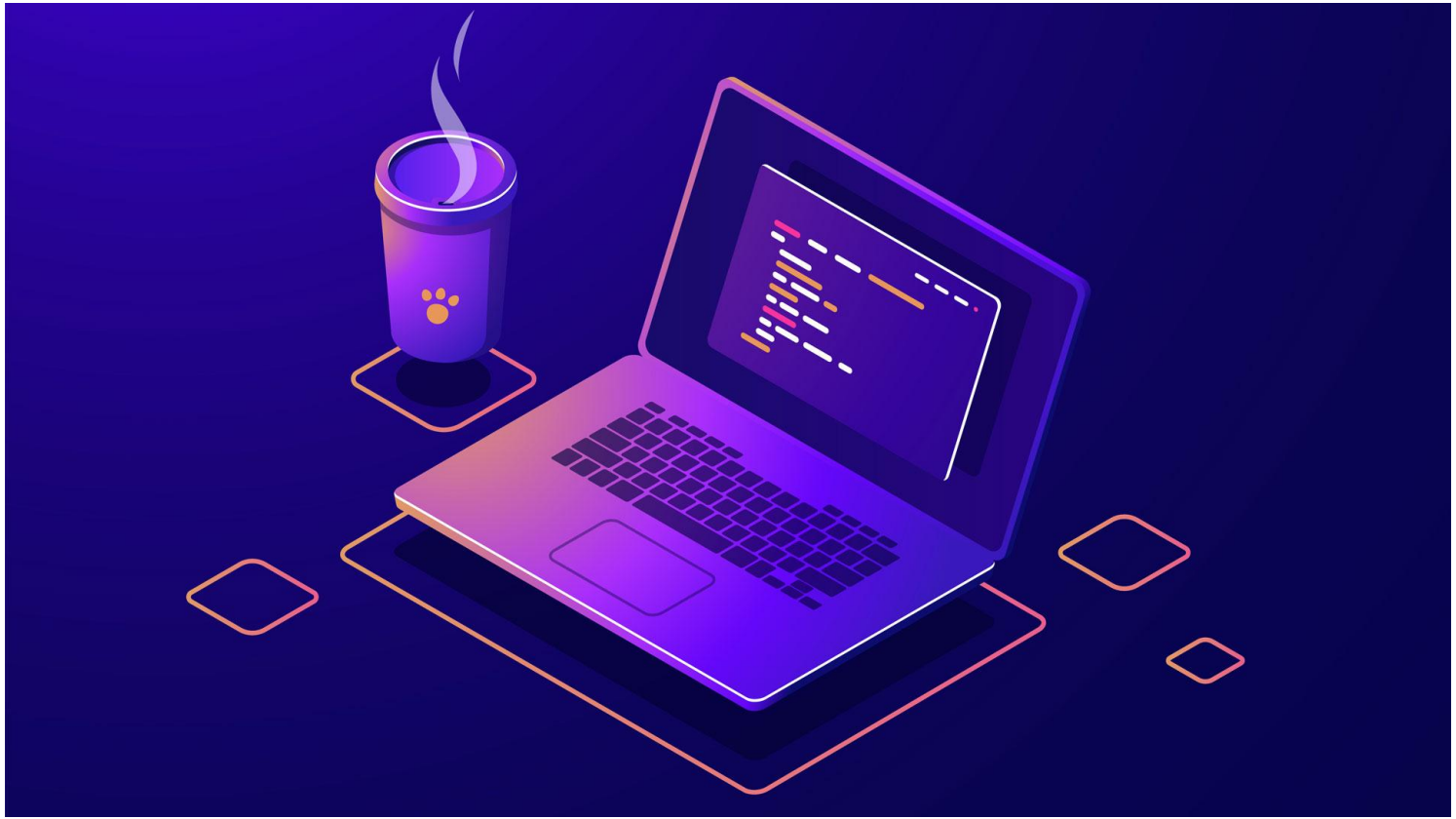
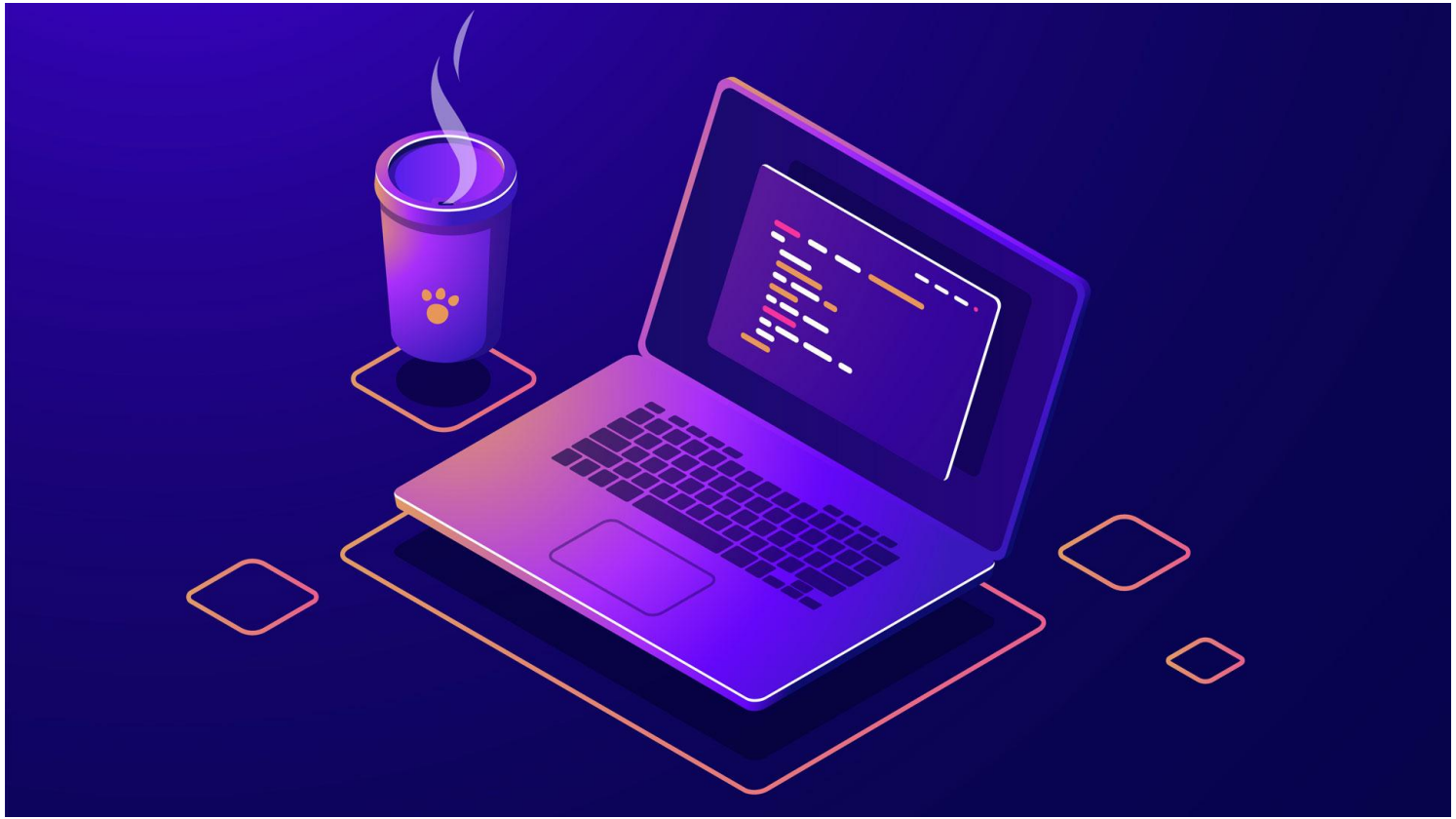
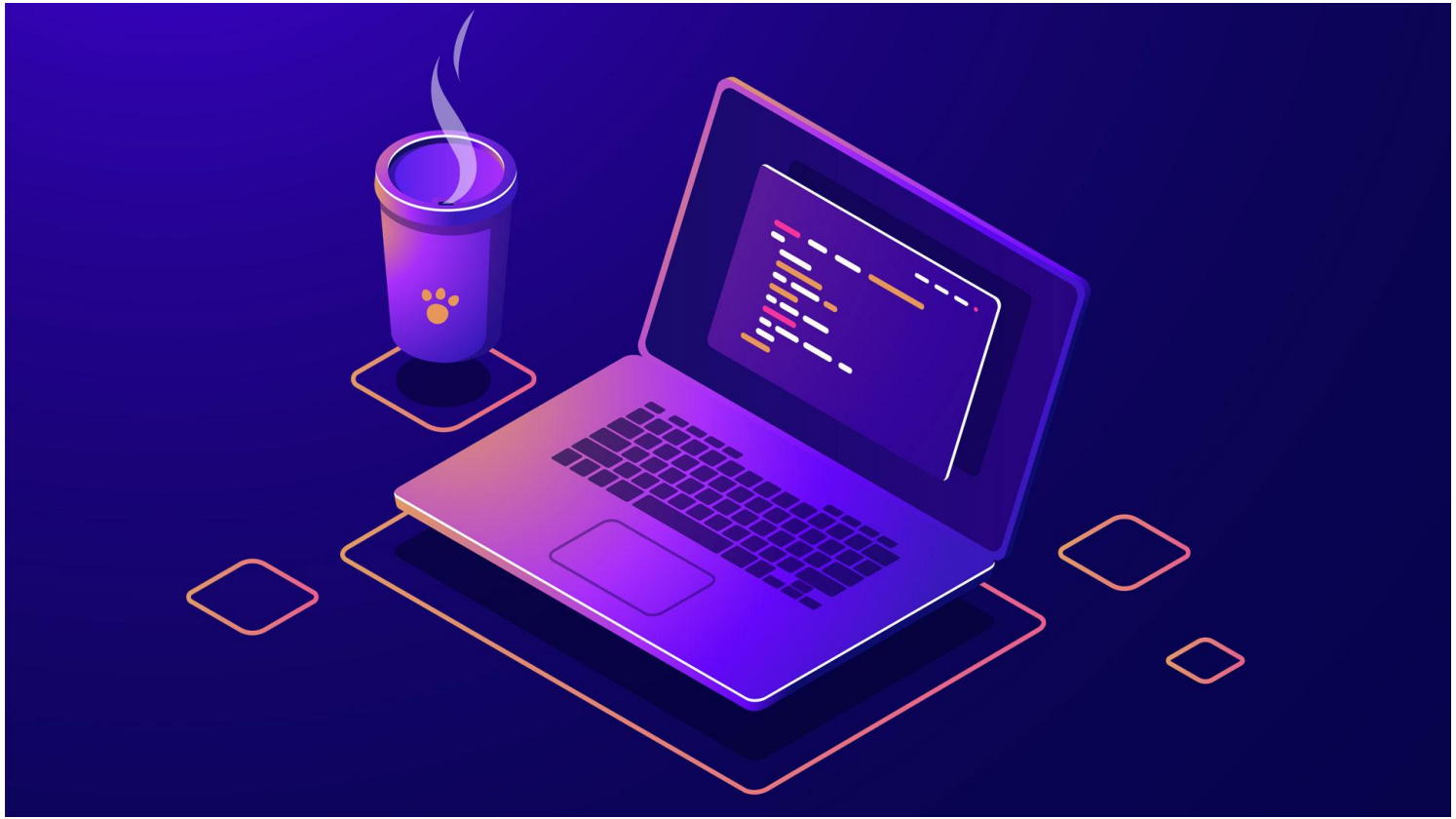
OpenMP Problems



Problem 1



Problem 1 - Point B



Problem 1

Given the following sequential code to count the number of times a value **key** appears in vector **a**

```
#define N 131072
long count_key(long Nlen, long *a, long key) {
    long count = 0;
    for (int i=0; i<Nlen; i++)
        if(a[i]==key) count++;
    return count;
}

int main() {
   long a[N], key = 42, nkey=0;
   for (long i=0;i<N;i++) a[i] = random()%N;
      a[N%43]=key; a[N%73]=key; a[N%3]=key;
   nkey = count_key(N, a, key);     // count key sequentially
   nkey = count_iter(N, a, key);    // count key in a using an iterative decomposition
   nkey = count_recur(N, a, key);   // count key in a with divide and conquer
}
```

Problem 1 - Point B

(a) Write a parallel OpenMP version using an iterative task decomposition (`count_iter`), in which as many tasks as threads are generated.

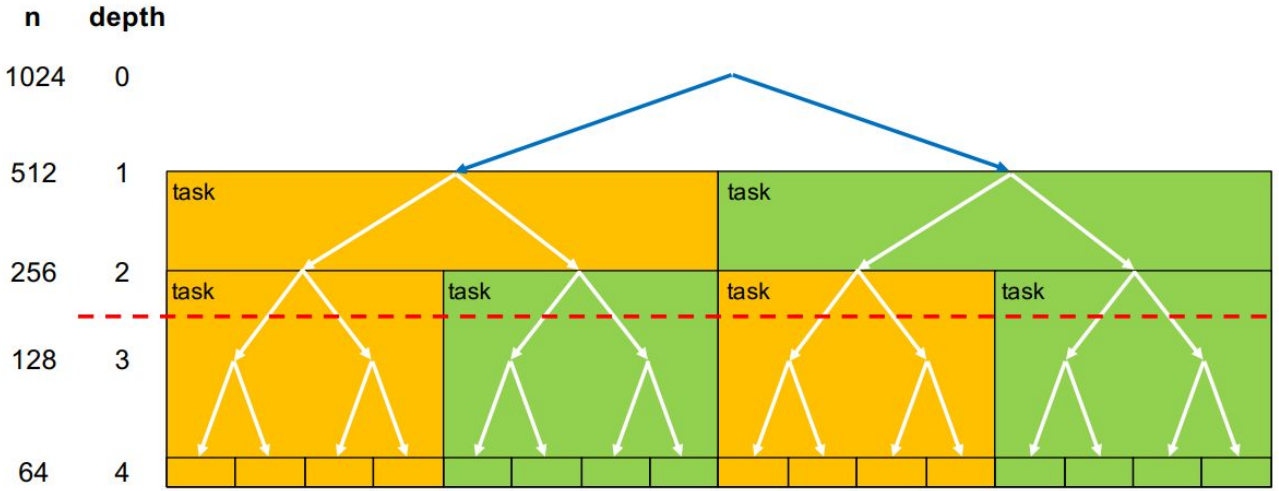
(b) Write a parallel OpenMP version using a recursive task decomposition (divide and conquer, `count_recur`). The implementation should take into account the overhead due to task creation, limiting their creation once a certain level in the recursive tree is reached.

Recursive Task Decomposition

In recursive task decomposition strategies one can control task granularity by controlling recursion levels where tasks are generated (cut-off control).

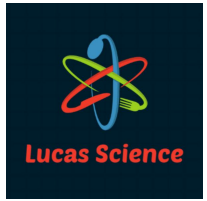
Recursive Task Decomposition

Tree strategy with **depth recursion control**



Instructor Social Media

Youtube: Lucas Science



Instagram: lucaasbazilio

Twitter: lucasebazilio