

- Coding *independent threads* can speed up the running of programs. Let us conduct an experiment to experience the features and skill of *multithreading*. You need to *code two different styles of programs* to perform a “matrix multiplication” in the same programming language (e.g. C language), as follows.

$$C_{35 \times 35} = A_{35 \times 60} \times B_{60 \times 35}$$

$$[a_{ij}] = 3.5i - 6.6j, [b_{ij}] = 6.6 + 8.8i - 3.5j$$

- ◆ In the *first* program, you just code it following the *traditional for-looping* skill.
- ◆ In the *second* program, you need trying to code it by using the new *multithreading* skill.

Q1: Point out the *major parts* in the threaded program to highlight its differences with *for-loops*.

for迴圈的運行方式是循環執行，執行緒是同時動作，像是Rows的執行緒就是每一列的計算同時動作；Cells則是每一個元素的計算同時動作。

在程式中的顯著差異就是在執行緒的寫法中，要創建執行緒，`pthread_t threads[]`和`pthread_create`函式可以用來建立新的執行緒，執行緒在建立之後，就會以同時動作的方式執行。再以`pthread_join`等待所有執行緒的結束，結束後將所有同時計算出的結果組合在一起輸出。

Q2: Record your *experimental results* at least 3 rounds execution in the below table, and state how you can count the running time of programs in ms.

Coding Skill	Lines of Code	Execution Time (ms)			Average Execution Time
		1-round	2-round	3-round	
【A】 <i>For-loops</i>	43	8302	8173	8207	8227.3
【B1】 <i>by cells</i> <i>Multithread</i>	77	6761	6687	6717	6721.6
Differences 【A - B1】	-34	1585	1486	1490	1520.3
【B2】 <i>by rows</i> <i>Multithread</i>	57	7570	7717	7397	7561.3
Differences 【A - B2】	-14	732	456	810	666

Q3: State your discovering and comments on this exercise of coding threaded programs.

完成這次的作業之後，我學到了如何寫執行緒的程式，也驚覺到，原來寫一個需要重複動作的程式，除了以前只會的for迴圈之外，還能有更好的做法，就是讓它同時執行。但畢竟這是我第一次寫執行緒的程式，在實作的過程中也是跌跌撞撞。

這次的作業需要寫三個程式，一開始，我是從for迴圈的程式下手，畢竟是相對簡單的一個部分，所以很快做完了。再來，我選擇做Rows的執行緒程式，也就是同時執行每一列的計算，我發現且參考了網路上的教學，使用pthread的函式庫，可以輕鬆地建立執行緒來做計算，再將A和B矩陣改上作業所要求的aij與bij算法就行，與for迴圈比較呢，是確實比較快，但我是測試到使用200*200的矩陣之後才比較看得出速度的差異，可以差到快一秒（我使用time.h寫入程式直接計算main()的執行時間）。

接著，來到了Cells的部分，每個執行緒計算一個元素，以200*200的矩陣來看，就是同時進行200*200條執行緒（雖然執行緒不會同時建構，但還是這樣講比較好懂）！除了看起來就比Rows難之外，其程式的寫法和Rows比較，確實複雜許多，原本在Rows的寫法，我只需考慮A矩陣的列向量，再乘上B矩陣的每一行就好，因為是每條執行緒算一個列，但到了Cells變成A矩陣列和行都要考慮，再去乘上對應的B矩陣元素，光是這點，在相乘時就複雜很多了，甚至還要將值先丟進暫存的data裡，所以Cells的部分我與同學討論了很久。

而在我兢兢業業地以為我終於做完的時候，我按下了執行，結果執行時間需要快17秒…，原本以為是我用200*200的矩陣相乘會產生太多的執行緒導致overhead過高，但我改成10*10測試，一樣跑得比for迴圈和Rows還慢，因為在執行初期會卡頓很久才顯示出答案，所以我又反覆修改了很久，但還是找不到原因。

結果，就在我一籌莫展之際，想說放在別的編譯與執行環境跑看看好了，就把我的程式傳給同學，讓他在他的電腦幫我試看看（我們都是用Dev C++），竟然就奇蹟般地能用！完全不會卡頓，執行時間只要6秒左右，比for迴圈快了一秒多，也比Rows快了快一秒，於是我才確定原來是環境問題，而這問題也在我重灌Dev C++之後消失了…，可能是我以前有改到什麼環境參數吧。

總之，我其實滿感謝有這次機會，讓我學習到了執行緒程式的寫法，相較於一直循環的for迴圈動作方式，執行緒能同時處理多個計算，讓程式執行的效率變好，將CPU的計算能力最大化，這與上課聽老師講理論完全是不同的體驗，果然凡事都是要親自做過才會懂呢，而要不是這次要交這份作業，我可能一輩子也不會碰到這塊領域，所以我很珍惜這次機會。

* Note that you need to take some pictures of the running screen on LCD, and appendix your two *source codes* in your report.

For-loops :

```
#include <iostream>
#include <cstdlib>
#include <time.h>

using namespace std;

// maximum size of matrix
#define MAX 200

void multi(int C[MAX][MAX],int A[MAX][MAX],int B[MAX][MAX]) {
    for(int i = 0; i < MAX; i++)
        for (int j = 0; j < MAX; j++) {
            C[i][j] = 0;
            for (int k = 0; k < MAX; k++) {
                A[i][k] = 3.5 *i - 6.6 *k;
                B[k][j] = 6.6+8.8 *k - 3.5 *j;
                C[i][j] += A[i][k] * B[k][j];
            }
        }
}

// Main Code
int main() {
    clock_t start, end;
    start = clock();
    int A[MAX][MAX];
    int B[MAX][MAX];
    int C[MAX][MAX];
    multi(C, A, B);
    cout <<"\n"
    <<"Multiplication of A and B:"<<"\n";
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << C[i][j]<<"\t";
        cout <<"\n";
    }
    end = clock();

    //Displaying the execution time
    double diff = end - start; // ms
    cout <<"\n"<<"execution time:"<<"\n"<<diff<<" ms";
    return 0;
}
```

```

for loop.cpp row.cpp cells.cpp
1 #include <iostream>
2 #include <cstdlib>
3 #include <time.h>
4
5 using namespace std;
6
7 // maximum size of matrix
8 #define MAX 200
9
10 void multi(int C[MAX][MAX], int A[MAX][MAX], int B[MAX][MAX]) {
11     for(int i = 0; i < MAX; i++)
12         for(int j = 0; j < MAX; j++) {
13             C[i][j] = 0;
14             for(int k = 0; k < MAX; k++) {
15                 A[i][k] = 3.5 * i - 6.6 * k;
16                 B[k][j] = 6.6 * 8 * k - 3.5 * j;
17                 C[i][j] += A[i][k] * B[k][j];
18             }
19         }
20 }
21
22 // Main Code
23 int main() {
24     clock_t start, end;
25     start = clock();
26     int A[MAX][MAX];
27     int B[MAX][MAX];
28     multi(A, B);
29     cout << "\n";
30     // Multiplication of A and B: << "\n";
31     for(int i = 0; i < MAX; i++) {
32         for(int j = 0; j < MAX; j++)
33             cout << C[i][j] << "\t";
34         cout << "\n";
35     }
36     end = clock();
37
38     //Displaying the execution time
39     double diff = end - start; // ms
40     cout << "\n" << "execution time: " << diff << " ms";
41     return 0;
42 }

```

execution time:
8302 ms

Cells Multithread :

```

#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
#include<stdlib.h>
#include <time.h>
#include <bits/stdc++.h>
using namespace std;
#define MAX 200

```

//Each thread computes single element in the resultant matrix

```

void *mult(void* arg) {
    int *data = (int *)arg;
    int k = 0, i = 0;

    int x = data[0];
    for (i = 1; i <= x; i++)
        k += data[i]*data[i+x];

    int *p = (int*)malloc(sizeof(int));
    *p = k;

    pthread_exit(p);
}

```

// Main Code

```

int main() {
    clock_t start, end;
    start = clock();

    int A[MAX][MAX];
    int B[MAX][MAX];
    int max = MAX*MAX;
    int i,j,k;

    //declaring array of threads of size MAX*MAX
    pthread_t *threads;
    threads = (pthread_t*)malloc(max*sizeof(pthread_t));

    int count = 0;
    int* data = NULL;
    for (i = 0; i < MAX; i++)
        for (j = 0; j < MAX; j++) {
            A[i][j] = 3.5 * i - 6.6 * j;
            B[i][j] = 6.6 + 8.8 * i - 3.5 * j;

            data = (int *)malloc((max)*sizeof(int));
            data[0] = MAX;

            for (k = 0; k < MAX; k++) {
                data[k+1] = A[i][k];
                data[k+MAX+1] = B[k][j];
            }

            //creating threads
            pthread_create(&threads[count++], NULL, mult, (void*)(data));
        }

    cout << "\n"
    << "Multiplication of A and B:" << "\n";

    for (i = 0; i < max; i++) {
        void *k;
        //join threads
        pthread_join(threads[i], &k);
        int *p = (int *)k;
        printf("%d ", *p);
        if ((i + 1) % MAX == 0)
            printf("\n");
    }
    end = clock();

    //Displaying the execution time
    double diff = end - start; // ms
    cout << "\n" << "execution time:" << "\n" << diff << " ms";
    return 0;
}

```

```

1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5 #include <time.h>
6 #include <bits/stdc++.h>
7 using namespace std;
8 #define MAX 200
9
10 //Each thread computes single element in the resultant matrix
11 void *multi(void* arg) {
12     int *data = (int *)arg;
13     int k = 0, i = 0;
14
15     int x = data[0];
16     for (i = 1; i <= x; i++)
17         k += data[i] * data[i+x];
18
19     int *p = (int *)malloc(sizeof(int));
20     *p = k;
21     pthread_exit(p);
22 }
23
24 // Main Code
25 int main() {
26     clock_t start, end;
27     start = clock();
28
29     int A[MAX][MAX];
30     int B[MAX][MAX];
31     int max = MAX * MAX;
32     int i, j, k;
33
34     //declaring array of threads of size MAX*MAX
35     pthread_t *threads;
36     threads = (pthread_t *)malloc(max * sizeof(pthread_t));
37
38     int count = 0;
39     int *data = NULL;
40
41     for (i = 0; i < MAX; i++)
42         for (j = 0; j < MAX; j++) {
43             A[i][j] = 3.5 * i - 6.6 * j;
44             B[i][j] = 6.6 * 8.8 * i - 3.5 * j;
45
46             data = (int *)malloc(max * sizeof(int));
47         }
48     }

```

```

C:\Users\Public\PeterVu\cells.exe
9832 520590 489506 456363 420991 383454 343401 300189 256147 207986 158909 105795 50737 -7192 -67491 -130766 -196633 -28
6497 -337399 -413099 -489922 -571528 -655312 -742646 -832624 -926258 -1022691 -1123827 -1226340 -1334199 -1443521 -15582
39 -1675501 -1796860 -1921269 -2049881 -2181064 -2317931 -2455827 -2599881 -2746887 -2898349 -3053257 -3212680 -3375684
3544320 -3543200 -3892098 -4071490 -4256932 -4445928 -4639746 -4837609 -5040274 -5247186 -5460044 -5675696 -5897793 -61
22985 -6354423 -6590173 -6830966 -7076536 -7327064 -7582636 -7844336 -8109561 -8381347 -8657172 -8939226 -9226482 -95188
70 -9816899 -10119909 -10428893 -10744055 -11064114 -11389741 -11721728 -12058200 -12402532 -12750318 -13106376 -1346567
0 -1383636 -14206062 -14585508 -14969794 -15361878 -15763206 -16163548 -16572169 -16990190 -17411207 -17842089 -1827721
7 -18720492 -19168367 -19622999 -20086103 -20557207 -21031284 -21516020 -22003379 -22501929 -23004879 -23516236 -240323
0 -24558852 -25088752 -25630370 -26174522 -26730725 -27289047 -27860017 -28434409 -29019599 -29608512 -30209397 -3081301
3 -31429897 -32048744 -32681166 -33315070 -33963212 -34614166 -35277442 -35943804 -36623971 -36647925
4176 9667 17123 24419 33581 43855 54527 66257 79586 91909 106424 121096 137236 152626 169745 187338 204823 222880 242107
259997 280073 298630 318724 337965 357922 378177 397760 417062 437898 45702 476943 495408 514751 532977 551863 569516
586478 602928 620098 635163 650851 664571 678434 691001 703454 714496 724695 733063 741847 748320 754936 759258 762922 7
65178 766479 766256 764363 760606 756830 749914 742338 732609 721355 708647 694077 677937 659233 638698 617065 592484 56
6709 537765 507385 474348 439899 402678 362955 320478 276781 229371 180644 127865 73607 16022 -43458 -106381 -171875 -24
0914 -311427 -386208 -462716 -543950 -628338 -713792 -802855 -896099 -992116 -1092337 -1194419 -1301347 -1410232 -152454
1 -1640811 -1761753 -1885151 -2013337 -2144061 -2279515 -2417344 -2560389 -2706318 -2857334 -3011158 -3170129 -3332030 -
3499593 -3669926 -3845785 -4024668 -4209005 -4396848 -4590188 -4786879 -4989062 -5194783 -5406484 -5621603 -5842487 -606
7175 -6297424 -6531933 -6772218 -7016528 -7266542 -7520835 -7781294 -8045952 -8316481 -8591726 -8872507 -9158434 -945028
3 -9746964 -10049430 -10357047 -10670884 -10990343 -11314626 -11644005 -11981114 -12324035 -12671244 -13023872 -13384583
-13751102 -14122113 -14500928 -14883783 -15275223 -15670104 -16073947 -16481961 -16894864 -17318868 -17748213 -18181842
-18624451 -19070811 -19527065 -19986338 -20455855 -20929293 -21412423 -21899139 -22396064 -22896931 -23408086 -23922571
-24448390 -24976675 -25516618 -26060101 -26614610 -27172257 -27741514 -28314239 -28898693 -29485923 -30086059 -30687976
-31303097 -31921244 -32551684 -33165083 -33831424 -34480627 -35143131 -35807726 -36487113 -37167101
execution time:
6761 ms
-----
Process exited after 6.913 seconds with return value 0
請按任意鍵繼續 . . .

```

```

execution time:
6761 ms
-----

```

Rows Multithread :

```

#include <bits/stdc++.h>
#include <time.h>

```

```
using namespace std;
```

```

// maximum size of matrix
#define MAX 200

```

```

int A[MAX][MAX];
int B[MAX][MAX];
int C[MAX][MAX];
int step_i = 0;

```

```

void* multi(void* arg) {
    int core = step_i++;

    // Each thread computes 1/MAX of matrix multiplication
    for (int i = core * MAX / MAX; i < (core + 1) * MAX / MAX; i++)
        for (int j = 0; j < MAX; j++)
            for (int k = 0; k < MAX; k++) {
                A[i][k] = 3.5 * i - 6.6 * k;
                B[k][j] = 6.6 * 8.8 * k - 3.5 * j;
                C[i][j] += A[i][k] * B[k][j];
            }
}

```

```
// Main Code
```

```
int main() {
    clock_t start, end;
    start = clock();
    //creating threads
    pthread_t threads[MAX];

    for (int i = 0; i < MAX; i++) {
        int* p;
        pthread_create(&threads[i], NULL, multi, (void*)(p));
    }

    for (int i = 0; i < MAX; i++)
        //join threads
        pthread_join(threads[i], NULL);

    // Displaying the multiplication matrix
    cout << "\n"
        << "Multiplication of A and B:" << "\n";
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << C[i][j] << " ";
        cout << "\n";
    }
    end = clock();

    //Displaying the execution time
    double diff = end - start; // ms
    cout << "\n" << "execution time:" << "\n" << diff << " ms";
    return 0;
}
```

```
for loop.cpp row.cpp cells.cpp
1 #include <bits/stdc++.h>
2 #include <time.h>
3
4 using namespace std;
5
6 // maximum size of matrix
7 #define MAX 200
8
9 int A[MAX][MAX];
10 int B[MAX][MAX];
11 int C[MAX][MAX];
12 int step_i = 0;
13
14 void* multi(void* arg) {
15     int core = step_i++;
16
17     // Each thread computes 1/MAX of matrix multiplication
18     for (int i = core * MAX / MAX; i < (core + 1) * MAX / MAX; i++)
19         for (int j = 0; j < MAX; j++)
20             for (int k = 0; k < MAX; k++) {
21                 A[i][k] = 3.5 * i - 6.6 * k;
22                 B[k][j] = 6.6 * 8.8 * k - 3.5 * j;
23                 C[i][j] += A[i][k] * B[k][j];
24             }
25 }
26
27 // Main Code
28 int main() {
29     clock_t start, end;
30     start = clock();
31     //creating threads
32     pthread_t threads[MAX];
33
34     for (int i = 0; i < MAX; i++) {
35         int* p;
36         pthread_create(&threads[i], NULL, multi, (void*)(p));
37     }
38
39     for (int i = 0; i < MAX; i++)
40         //join threads
41         pthread_join(threads[i], NULL);
42
43     // Displaying the multiplication matrix
44     cout << "\n"
45         << "Multiplication of A and B:" << "\n";
46     for (int i = 0; i < MAX; i++) {
47         for (int j = 0; j < MAX; j++)
```

```
C:\Users\Peter\Public\row.cpp
-35110730 -35333338 -35551140 -35768854 -35981156 -36194646 -36401979 -36608069 -36812408 -36915498 -37016200 -37115924 -37213956 -37310677 -37406494 -37500804 -37593116 -37682428 -37769240 -37853052 -37934364 -38012676 -38088488 -38161200 -38230412 -38296624 -38359236 -38417848 -38472960 -38525072 -38573684 -38618396 -38658808 -38695620 -38728432 -38761844 -38786256 -38811268 -38837280 -38863292 -38889304 -38915316 -38941328 -38967340 -38993352 -39019364 -39045376 -39071388 -39097400 -39123412 -39149424 -39175436 -39201448 -39227460 -39253472 -39279484 -39305496 -39331508 -39357520 -39383532 -39409544 -39435556 -39461568 -39487580 -39513592 -39539604 -39565616 -39591628 -39617640 -39643652 -39669664 -39695676 -39721688 -39747700 -39773712 -39799724 -39825736 -39851748 -39877760 -39903772 -39929784 -39955796 -39981808 -40007820 -40033832 -40059844 -40085856 -40111868 -40137880 -40163892 -40189904 -40215916 -40241928 -40267940 -40293952 -40319964 -40345976 -40371988 -40397996 -40423996 -40449996 -40475996 -40501996 -40527996 -40553996 -40579996 -40605996 -40631996 -40657996 -40683996 -40709996 -40735996 -40761996 -40787996 -40813996 -40839996 -40865996 -40891996 -40917996 -40943996 -40969996 -40995996 -41021996 -41047996 -41073996 -41099996 -41125996 -41151996 -41177996 -41203996 -41229996 -41255996 -41281996 -41307996 -41333996 -41359996 -41385996 -41411996 -41437996 -41463996 -41489996 -41515996 -41541996 -41567996 -41593996 -41619996 -41645996 -41671996 -41697996 -41723996 -41749996 -41775996 -41801996 -41827996 -41853996 -41879996 -41905996 -41931996 -41957996 -41983996 -42009996 -42035996 -42061996 -42087996 -42113996 -42139996 -42165996 -42191996 -42217996 -42243996 -42269996 -42295996 -42321996 -42347996 -42373996 -42399996 -42425996 -42451996 -42477996 -42503996 -42529996 -42555996 -42581996 -42607996 -42633996 -42659996 -42685996 -42711996 -42737996 -42763996 -42789996 -42815996 -42841996 -42867996 -42893996 -42919996 -42945996 -42971996 -42997996 -43023996 -43049996 -43075996 -43101996 -43127996 -43153996 -43179996 -43205996 -43231996 -43257996 -43283996 -43309996 -43335996 -43361996 -43387996 -43413996 -43439996 -43465996 -43491996 -43517996 -43543996 -43569996 -43595996 -43621996 -43647996 -43673996 -43699996 -43725996 -43751996 -43777996 -43803996 -43829996 -43855996 -43881996 -43907996 -43933996 -43959996 -43985996 -44011996 -44037996 -44063996 -44089996 -44115996 -44141996 -44167996 -44193996 -44219996 -44245996 -44271996 -44297996 -44323996 -44349996 -44375996 -44401996 -44427996 -44453996 -44479996 -44505996 -44531996 -44557996 -44583996 -44609996 -44635996 -44661996 -44687996 -44713996 -44739996 -44765996 -44791996 -44817996 -44843996 -44869996 -44895996 -44921996 -44947996 -44973996 -44999996 -45025996 -45051996 -45077996 -45103996 -45129996 -45155996 -45181996 -45207996 -45233996 -45259996 -45285996 -45311996 -45337996 -45363996 -45389996 -45415996 -45441996 -45467996 -45493996 -45519996 -45545996 -45571996 -45597996 -45623996 -45649996 -45675996 -45701996 -45727996 -45753996 -45779996 -45805996 -45831996 -45857996 -45883996 -45909996 -45935996 -45961996 -45987996 -46013996 -46039996 -46065996 -46091996 -46117996 -46143996 -46169996 -46195996 -46221996 -46247996 -46273996 -46299996 -46325996 -46351996 -46377996 -46403996 -46429996 -46455996 -46481996 -46507996 -46533996 -46559996 -46585996 -46611996 -46637996 -46663996 -46689996 -46715996 -46741996 -46767996 -46793996 -46819996 -46845996 -46871996 -46897996 -46923996 -46949996 -46975996 -46999996 -47025996 -47051996 -47077996 -47103996 -47129996 -47155996 -47181996 -47207996 -47233996 -47259996 -47285996 -47311996 -47337996 -47363996 -47389996 -47415996 -47441996 -47467996 -47493996 -47519996 -47545996 -47571996 -47597996 -47623996 -47649996 -47675996 -47701996 -47727996 -47753996 -47779996 -47805996 -47831996 -47857996 -47883996 -47909996 -47935996 -47961996 -47987996 -48013996 -48039996 -48065996 -48091996 -48117996 -48143996 -48169996 -48195996 -48221996 -48247996 -48273996 -48299996 -48325996 -48351996 -48377996 -48403996 -48429996 -48455996 -48481996 -48507996 -48533996 -48559996 -48585996 -48611996 -48637996 -48663996 -48689996 -48715996 -48741996 -48767996 -48793996 -48819996 -48845996 -48871996 -48897996 -48923996 -48949996 -48975996 -48999996 -49025996 -49051996 -49077996 -49103996 -49129996 -49155996 -49181996 -49207996 -49233996 -49259996 -49285996 -49311996 -49337996 -49363996 -49389996 -49415996 -49441996 -49467996 -49493996 -49519996 -49545996 -49571996 -49597996 -49623996 -49649996 -49675996 -49701996 -49727996 -49753996 -49779996 -49805996 -49831996 -49857996 -49883996 -49909996 -49935996 -49961996 -49987996 -50013996 -50039996 -50065996 -50091996 -50117996 -50143996 -50169996 -50195996 -50221996 -50247996 -50273996 -50299996 -50325996 -50351996 -50377996 -50403996 -50429996 -50455996 -50481996 -50507996 -50533996 -50559996 -50585996 -50611996 -50637996 -50663996 -50689996 -50715996 -50741996 -50767996 -50793996 -50819996 -50845996 -50871996 -50897996 -50923996 -50949996 -50975996 -50999996 -51025996 -51051996 -51077996 -51103996 -51129996 -51155996 -51181996 -51207996 -51233996 -51259996 -51285996 -51311996 -51337996 -51363996 -51389996 -51415996 -51441996 -51467996 -51493996 -51519996 -51545996 -51571996 -51597996 -51623996 -51649996 -51675996 -51701996 -51727996 -51753996 -51779996 -51805996 -51831996 -51857996 -51883996 -51909996 -51935996 -51961996 -51987996 -52013996 -52039996 -52065996 -52091996 -52117996 -52143996 -52169996 -52195996 -52221996 -52247996 -52273996 -52299996 -52325996 -52351996 -52377996 -52403996 -52429996 -52455996 -52481996 -52507996 -52533996 -52559996 -52585996 -52611996 -52637996 -52663996 -52689996 -52715996 -52741996 -52767996 -52793996 -52819996 -52845996 -52871996 -52897996 -52923996 -52949996 -52975996 -52999996 -53025996 -53051996 -53077996 -53103996 -53129996 -53155996 -53181996 -53207996 -53233996 -53259996 -53285996 -53311996 -53337996 -53363996 -53389996 -53415996 -53441996 -53467996 -53493996 -53519996 -53545996 -53571996 -53597996 -53623996 -53649996 -53675996 -53701996 -53727996 -53753996 -53779996 -53805996 -53831996 -53857996 -53883996 -53909996 -53935996 -53961996 -53987996 -54013996 -54039996 -54065996 -54091996 -54117996 -54143996 -54169996 -54195996 -54221996 -54247996 -54273996 -54299996 -54325996 -54351996 -54377996 -54403996 -54429996 -54455996 -54481996 -54507996 -54533996 -54559996 -54585996 -54611996 -54637996 -54663996 -54689996 -54715996 -54741996 -54767996 -54793996 -54819996 -54845996 -54871996 -54897996 -54923996 -54949996 -54975996 -54999996 -55025996 -55051996 -55077996 -55103996 -55129996 -55155996 -55181996 -55207996 -55233996 -55259996 -55285996 -55311996 -55337996 -55363996 -55389996 -55415996 -55441996 -55467996 -55493996 -55519996 -55545996 -55571996 -55597996 -55623996 -55649996 -55675996 -55701996 -55727996 -55753996 -55779996 -55805996 -55831996 -55857996 -55883996 -55909996 -55935996 -55961996 -55987996 -56013996 -56039996 -56065996 -56091996 -56117996 -56143996 -56169996 -56195996 -56221996 -56247996 -56273996 -56299996 -56325996 -56351996 -56377996 -56403996 -56429996 -56455996 -56481996 -56507996 -56533996 -56559996 -56585996 -56611996 -56637996 -56663996 -56689996 -56715996 -56741996 -56767996 -56793996 -56819996 -56845996 -56871996 -56897996 -56923996 -56949996 -56975996 -56999996 -57025996 -57051996 -57077996 -57103996 -57129996 -57155996 -57181996 -57207996 -57233996 -57259996 -57285996 -57311996 -57337996 -57363996 -57389996 -57415996 -57441996 -57467996 -57493996 -57519996 -57545996 -57571996 -57597996 -57623996 -57649996 -57675996 -57701996 -57727996 -57753996 -57779996 -57805996 -57831996 -57857996 -57883996 -57909996 -57935996 -57961996 -57987996 -58013996 -58039996 -58065996 -58091996 -58117996 -58143996 -58169996 -58195996 -58221996 -58247996 -58273996 -58299996 -58325996 -58351996 -58377996 -58403996 -58429996 -58455996 -58481996 -58507996 -58533996 -58559996 -58585996 -58611996 -58637996 -58663996 -58689996 -58715996 -58741996 -58767996 -58793996 -58819996 -58845996 -58871996 -58897996 -58923996 -58949996 -58975996 -58999996 -59025996 -59051996 -59077996 -59103996 -59129996 -59155996 -59181996 -59207996 -59233996 -59259996 -59285996 -59311996 -59337996 -59363996 -59389996 -59415996 -59441996 -59467996 -59493996 -59519996 -59545996 -59571996 -59597996 -59623996 -59649996 -59675996 -59701996 -59727996 -59753996 -59779996 -59805996 -59831996 -59857996 -59883996 -59909996 -59935996 -59961996 -59987996 -60013996 -60039996 -60065996 -60091996 -60117996 -60143996 -60169996 -60195996 -60221996 -60247996 -60273996 -60299996 -60325996 -60351996 -60377996 -60403996 -60429996 -60455996 -60481996 -60507996 -60533996 -60559996 -60585996 -60611996 -60637996 -60663996 -60689996 -60715996 -60741996 -60767996 -60793996 -60819996 -60845996 -60871996 -60897996 -60923996 -60949996 -60975996 -60999996 -61025996 -61051996 -61077996 -61103996 -61129996 -61155996 -61181996 -61207996 -61233996 -61259996 -61285996 -61311996 -61337996 -61363996 -61389996 -61415996 -61441996 -61467996 -61493996 -61519996 -61545996 -61571996 -61597996 -61623996 -61649996 -61675996 -61701996 -61727996 -61753996 -61779996 -61805996 -61831996 -61857996 -61883996 -61909996 -61935996 -61961996 -61987996 -62013996 -62039996 -62065996 -62091996 -62117996 -62143996 -62169996 -62195996 -62221996 -62247996 -62273996 -62299996 -62325996 -62351996 -62377996 -62403996 -62429996 -62455996 -62481996 -62507996 -62533996 -62559996 -62585996 -62611996 -62637996 -62663996 -62689996 -62715996 -62741996 -62767996 -62793996 -62819996 -62845996 -62871996 -62897996 -62923996 -62949996 -62975996 -62999996 -63025996 -63051996 -63077996 -63103996 -63129996 -63155996 -63181996 -63207996 -63233996 -63259996 -63285996 -63311996 -63337996 -63363996 -63389996 -63415996 -63441996 -63467996 -63493996 -63519996 -63545996 -63571996 -63597996 -63623996 -63649996 -63675996 -63701996 -63727996 -63753996 -63779996 -63805996 -63831996 -63857996 -63883996 -63909996 -63935996 -63961996 -63987996 -64013996 -64039996 -64065996 -64091996 -64117996 -64143996 -64169996 -64195996 -64221996 -64247996 -64273996 -64299996 -64325996 -64351996 -64377996 -64403996 -64429996 -64455996 -64481996 -64507996 -64533996 -64559996 -64585996 -64611996 -64637996 -64663996 -64689996 -64715996 -64741996 -64767996 -64793996 -64819996 -64845996 -64871996 -64897996 -64923996 -64949996 -64975996 -64999996 -65025996 -65051996 -65077996 -65103996 -65129996 -65155996 -65181996 -65207996 -65233996 -65259996 -65285996 -65311996 -65337996 -65363996 -65389996 -65415996 -65441996 -65467996 -65493996 -65519996 -65545996 -65571996 -65597996 -65623996 -65649996 -65675996 -65701996 -65727996 -65753996 -65779996 -65805996 -65831996 -65857996 -65883996 -65909996 -65935996 -65961996 -65987996 -66013996 -66039996 -66065996 -66091996 -66117996 -66143996 -66169996 -66195996 -66221996 -66247996 -66273996 -66299996 -66325996 -66351996 -66377996 -66403996 -66429996 -66455996 -66481996 -66507996 -66533996 -66559996 -66585996 -66611996 -66637996 -66663996 -66689996 -66715996 -66741996 -66767996 -66793996 -66819996 -66845996 -66871996 -66897996 -66923996 -66949996 -66975996 -66999996 -67025996 -67051996 -67077996 -67103996 -67129996 -67155996 -67181996 -67207996 -67233996 -67259996 -67285996 -67311996 -67337996 -67363996 -67389996 -67415996 -67441996 -67467996 -67493996 -67519996 -67545996 -67571996 -67597996 -67623996 -67649996 -67675996 -67701996 -67727996 -67753996 -67779996 -67805996 -67831996 -67857996 -67883996 -67909996 -67935996 -67961996 -67987996 -68013996 -68039996 -68065996 -68091996 -68117996 -68143996 -68169996 -68195996 -68221996 -68247996 -68273996 -68299996 -68325996 -68351996 -68377996 -68403996 -68429996 -68455996 -68481996 -68507996 -68533996 -68559996 -68585996 -68611996 -68637996 -68663996 -68689996 -68715996 -68741996 -68767996 -68793996 -68819996 -68845996 -68871996 -68897996 -68923996 -68949996 -68975996 -68999996 -69025996 -69051996 -69077996 -69103996 -69129996 -69155996 -69181996 -69207996 -69233996 -69259996 -69285996 -69311996 -69337996 -69363996 -69389996 -69415996 -69441996 -69467996 -69493996 -69519996 -69545996 -69571996 -69597996 -69623996 -69649996 -69675996 -69701996 -69727996 -69753996 -69779996 -69805996 -69831996 -69857996 -69883996 -69909996 -69935996 -69961996 -69987996 -70013996 -70039996 -70065996 -70091996 -70117996 -70143996 -70169996 -70195996 -70221996 -70247996 -70273996 -70299996 -70325996 -70351996 -70377996 -7
```