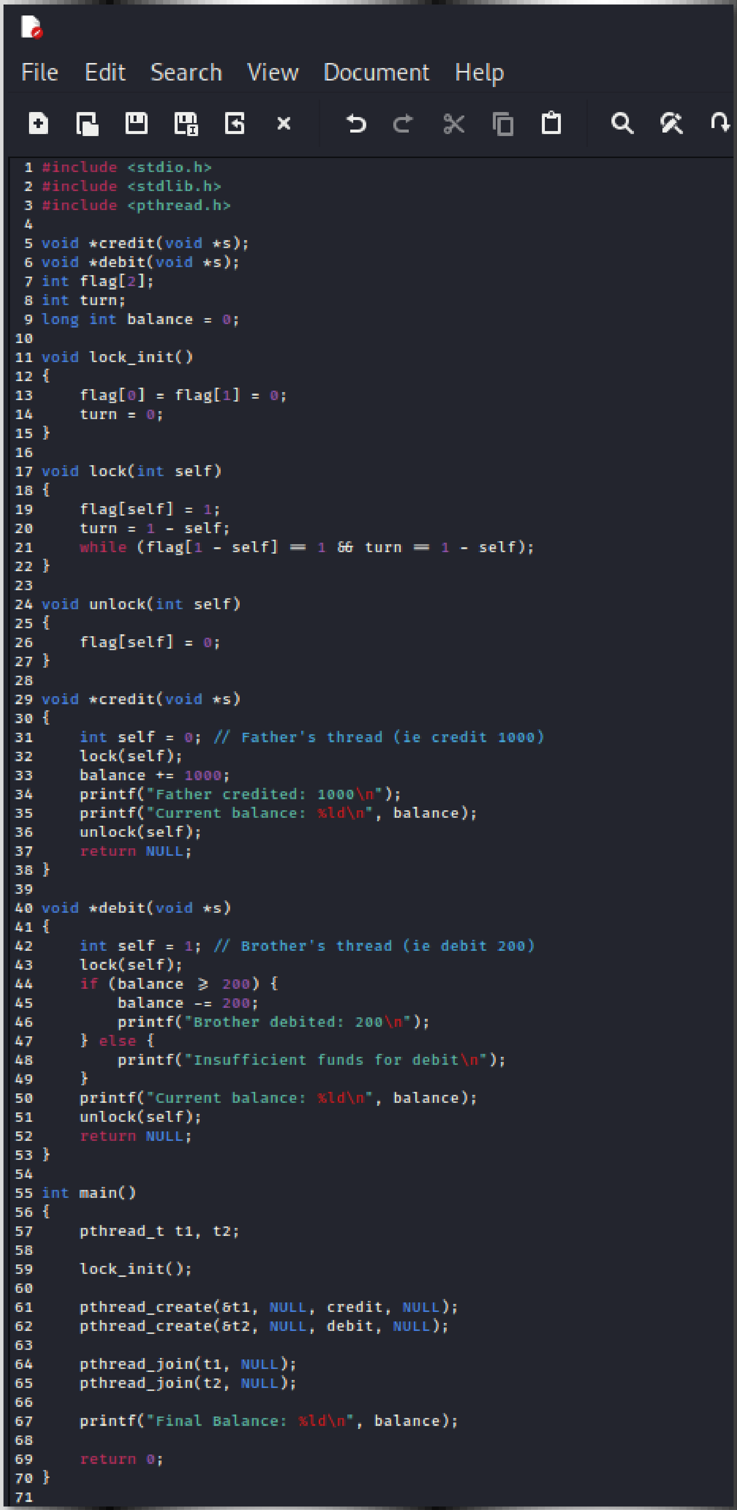
DIVYANSHU PATEL

23BAI1214

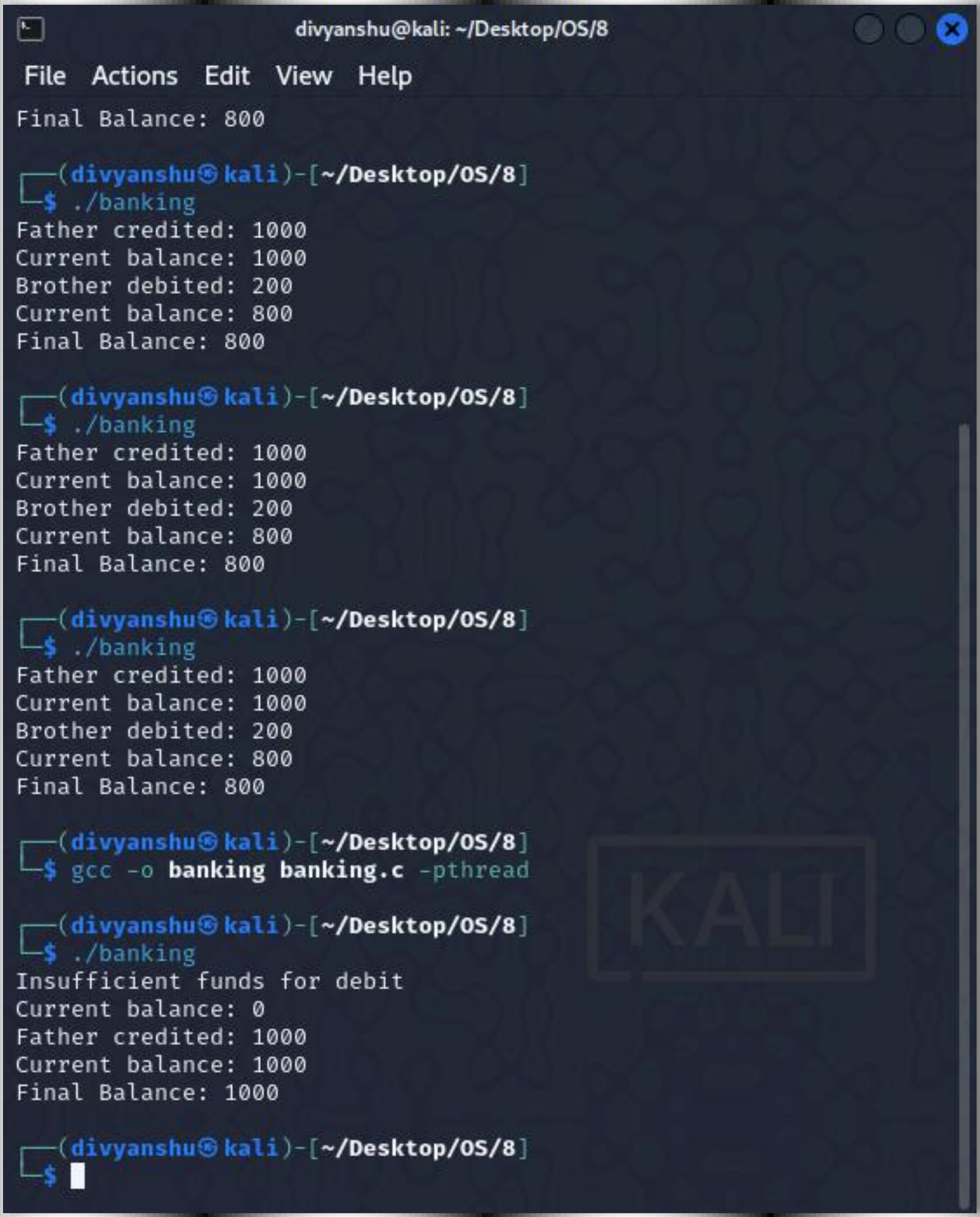
LAB-8-PETERSON'S ALGORITHM

GITHUB Link LAB 8 (divyanshupatel17)

1. Develop a minimal banking application for a single account. Father credits the account with 1000rs and brother debits the account with 200 rs using appropriate synchronization technique

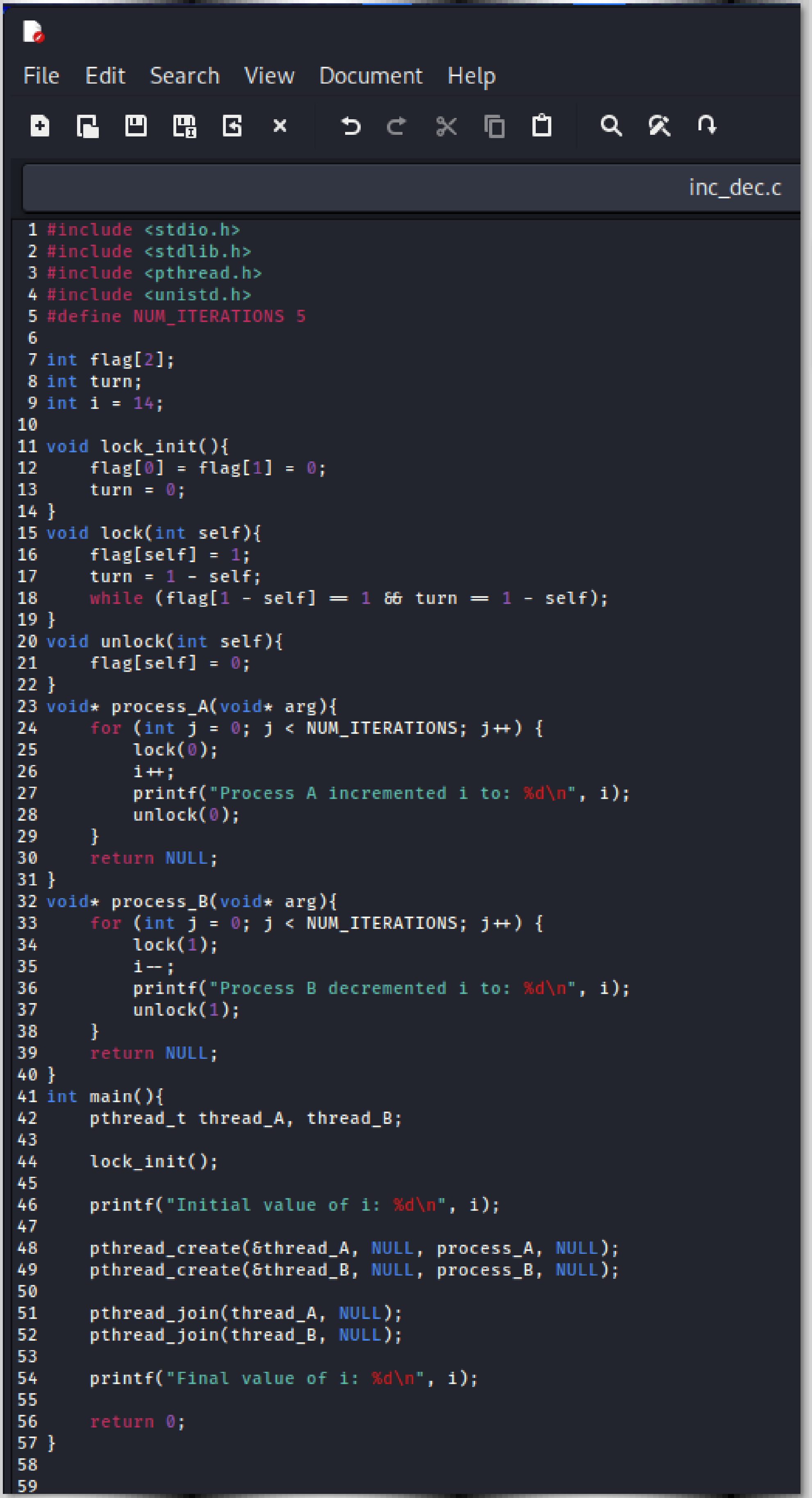


```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *credit(void *s);
void *debit(void *s);
int flag[2];
int turn;
long int balance = 0;

void lock_init()
{
    flag[0] = flag[1] = 0;
    turn = 0;
}

void lock(int self)
{
    flag[self] = 1;
    turn = 1 - self;
    while (flag[1 - self] == 1 && turn == 1 - self);
}

void unlock(int self)
{
    flag[self] = 0;
}

void *credit(void *s)
{
    int self = 0; // Father's thread (ie credit 1000)
    lock(self);
    balance += 1000;
    printf("Father credited: 1000\n");
    printf("Current balance: %ld\n", balance);
    unlock(self);
    return NULL;
}

void *debit(void *s)
{
    int self = 1; // Brother's thread (ie debit 200)
    lock(self);
    if (balance >= 200) {
        balance -= 200;
        printf("Brother debited: 200\n");
    } else {
        printf("Insufficient funds for debit\n");
    }
    printf("Current balance: %ld\n", balance);
    unlock(self);
    return NULL;
}

int main()
{
    pthread_t t1, t2;

    lock_init();

    pthread_create(&t1, NULL, credit, NULL);
    pthread_create(&t2, NULL, debit, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Final Balance: %ld\n", balance);

    return 0;
}

```



```
Final Balance: 800

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./banking
Father credited: 1000
Current balance: 1000
Brother debited: 200
Current balance: 800
Final Balance: 800

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./banking
Father credited: 1000
Current balance: 1000
Brother debited: 200
Current balance: 800
Final Balance: 800

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./banking
Father credited: 1000
Current balance: 1000
Brother debited: 200
Current balance: 800
Final Balance: 800

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ gcc -o banking banking.c -pthread

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./banking
Insufficient funds for debit
Current balance: 0
Father credited: 1000
Current balance: 1000
Final Balance: 1000

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ 
```

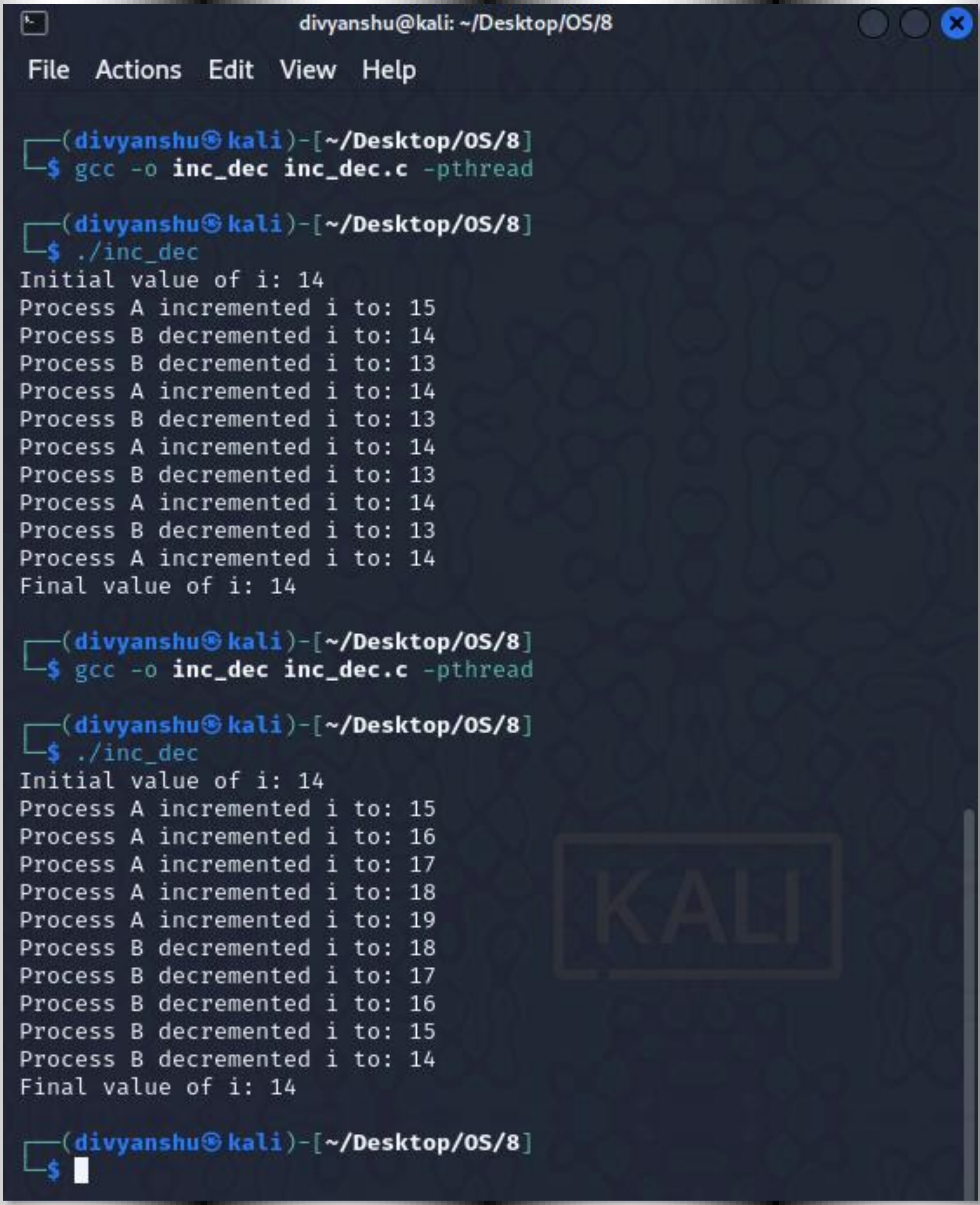
2.	Process A and B shares the value of i. its initial value is your registration number last two digits. Process A increments the value of i and B decrements the value of i



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 5

int flag[2];
int turn;
int i = 14;

void lock_init(){
    flag[0] = flag[1] = 0;
    turn = 0;
}
void lock(int self){
    flag[self] = 1;
    turn = 1 - self;
    while (flag[1 - self] == 1 && turn == 1 - self);
}
void unlock(int self){
    flag[self] = 0;
}
void* process_A(void* arg){
    for (int j = 0; j < NUM_ITERATIONS; j++) {
        lock(0);
        i++;
        printf("Process A incremented i to: %d\n", i);
        unlock(0);
    }
    return NULL;
}
void* process_B(void* arg){
    for (int j = 0; j < NUM_ITERATIONS; j++) {
        lock(1);
        i--;
        printf("Process B decremented i to: %d\n", i);
        unlock(1);
    }
    return NULL;
}
int main(){
    pthread_t thread_A, thread_B;

    lock_init();

    printf("Initial value of i: %d\n", i);

    pthread_create(&thread_A, NULL, process_A, NULL);
    pthread_create(&thread_B, NULL, process_B, NULL);

    pthread_join(thread_A, NULL);
    pthread_join(thread_B, NULL);

    printf("Final value of i: %d\n", i);

    return 0;
}


```

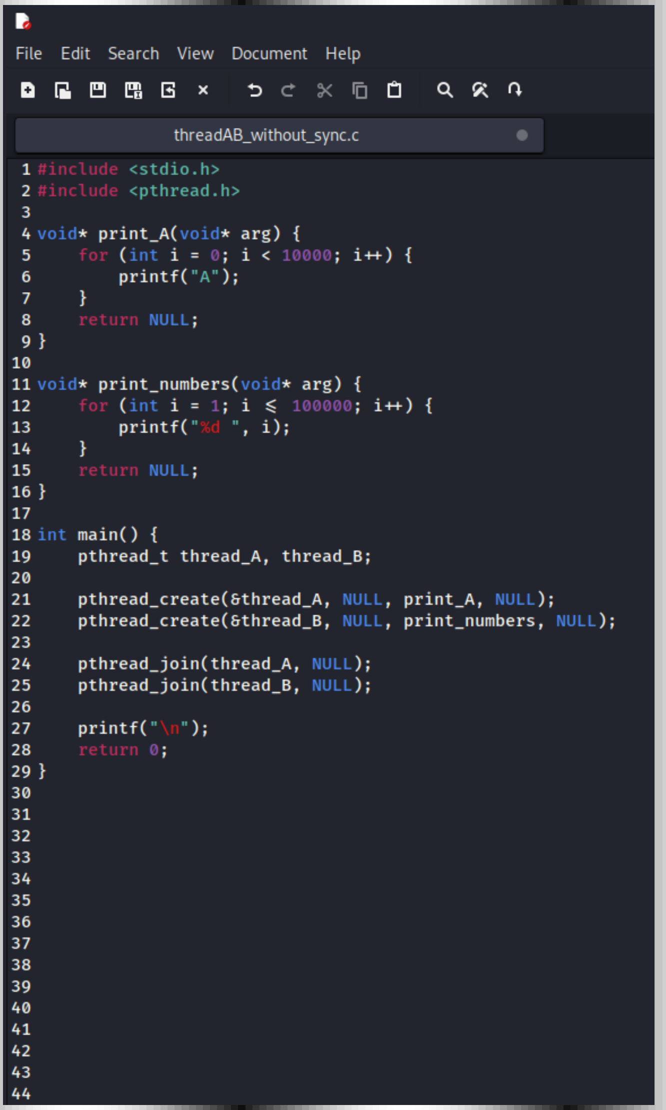
My Reg No:

23BAI1214

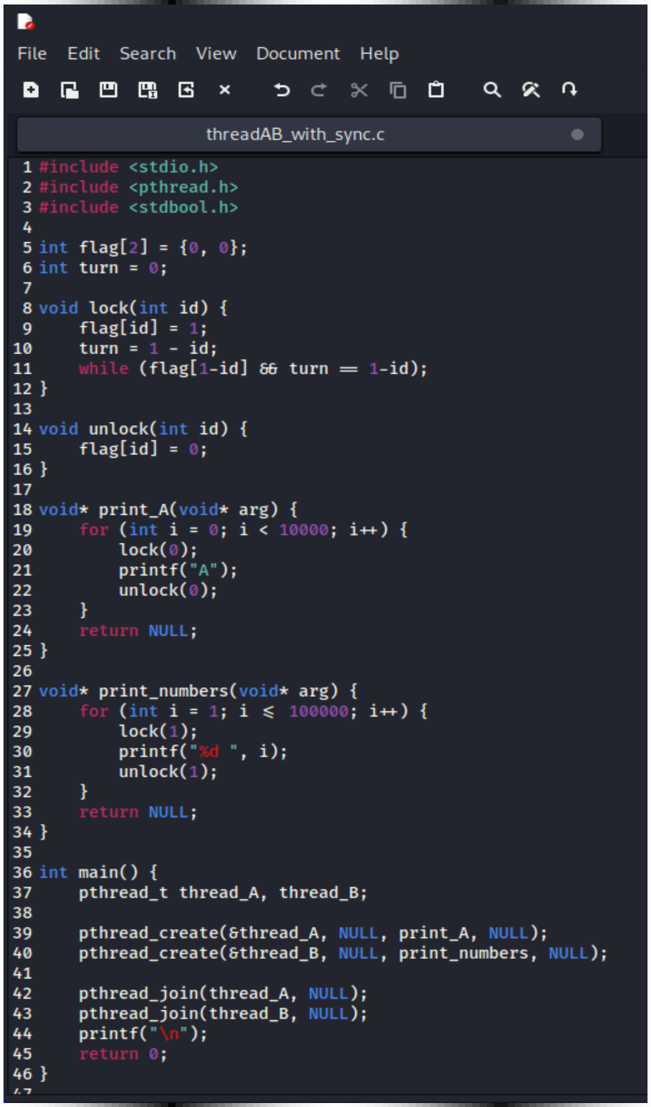


```
┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ gcc -o inc_dec inc_dec.c -pthread

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./inc_dec
Initial value of i: 14
Process A incremented i to: 15
Process B decremented i to: 14
Process B decremented i to: 13
Process A incremented i to: 14
Process B decremented i to: 13
Process A incremented i to: 14
Process B decremented i to: 13
Process A incremented i to: 14
Process B decremented i to: 13
Process A incremented i to: 14
Final value of i: 14

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ gcc -o inc_dec inc_dec.c -pthread

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./inc_dec
Initial value of i: 14
Process A incremented i to: 15
Process A incremented i to: 16
Process A incremented i to: 17
Process A incremented i to: 18
Process A incremented i to: 19
Process B decremented i to: 18
Process B decremented i to: 17
Process B decremented i to: 16
Process B decremented i to: 15
Process B decremented i to: 14
Final value of i: 14

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ 
```

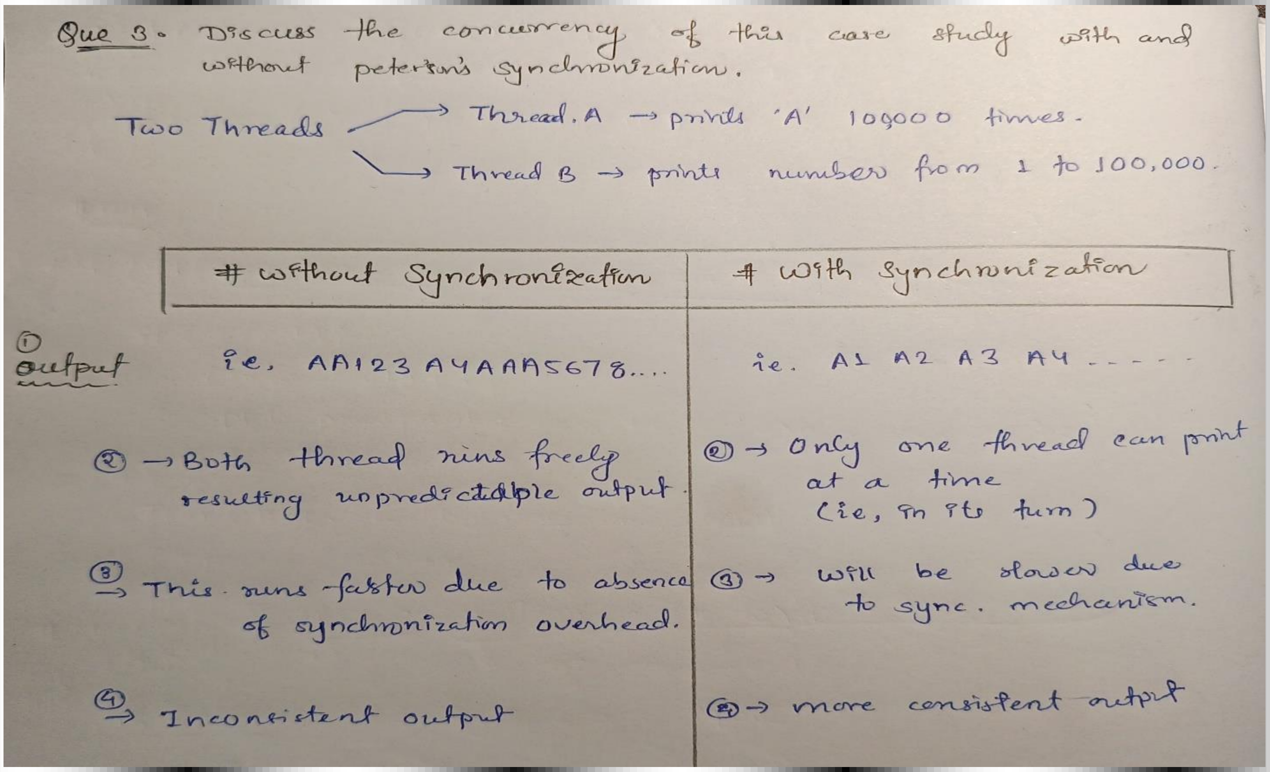
3. Create two threads A and B. One thread prints "A" 10000 times. Other thread B prints the i value from 1 to 100000. Discuss the concurrency of this case study with and without peter's synchronization and submit as a written assignment. along with corresponding code.



```c
#include <stdio.h>
#include <pthread.h>

void* print_A(void* arg) {
    for (int i = 0; i < 10000; i++) {
        printf("A");
    }
    return NULL;
}

void* print_numbers(void* arg) {
    for (int i = 1; i <= 100000; i++) {
        printf("%d ", i);
    }
    return NULL;
}

int main() {
    pthread_t thread_A, thread_B;

    pthread_create(&thread_A, NULL, print_A, NULL);
    pthread_create(&thread_B, NULL, print_numbers, NULL);

    pthread_join(thread_A, NULL);
    pthread_join(thread_B, NULL);

    printf("\n");
    return 0;
}
```



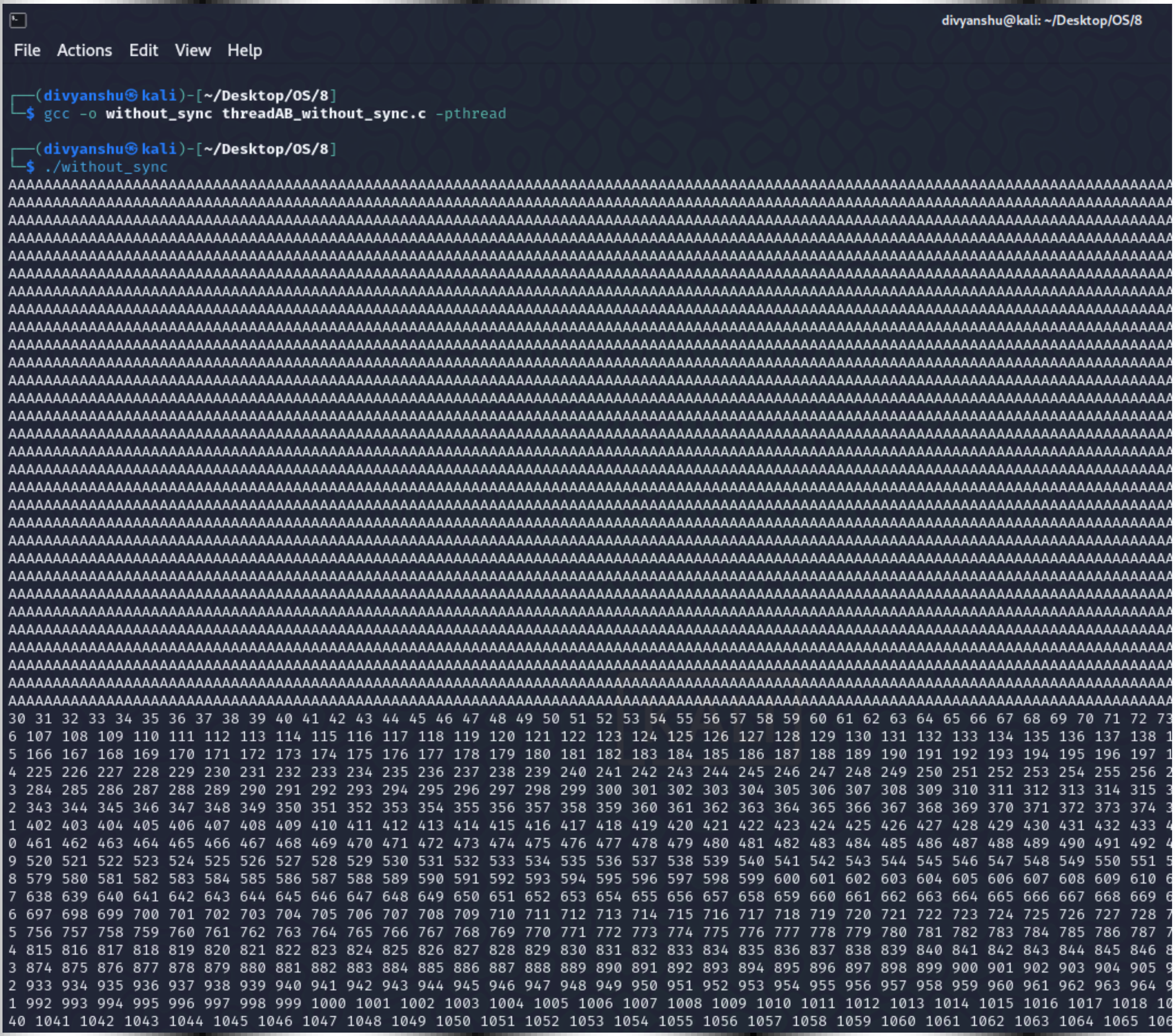
```c
#include <stdio.h>
#include <pthread.h>
#include <stdbool.h>

int flag[2] = {0, 0};
int turn = 0;

void lock(int id) {
    flag[id] = 1;
    turn = 1 - id;
    while (flag[1-id] && turn == 1-id);
}

void unlock(int id) {
    flag[id] = 0;
}

void* print_A(void* arg) {
    for (int i = 0; i < 10000; i++) {
        lock(0);
        printf("A");
        unlock(0);
    }
    return NULL;
}

void* print_numbers(void* arg) {
    for (int i = 1; i <= 100000; i++) {
        lock(1);
        printf("%d ", i);
        unlock(1);
    }
    return NULL;
}

int main() {
    pthread_t thread_A, thread_B;

    pthread_create(&thread_A, NULL, print_A, NULL);
    pthread_create(&thread_B, NULL, print_numbers, NULL);

    pthread_join(thread_A, NULL);
    pthread_join(thread_B, NULL);
    printf("\n");
    return 0;
}
```



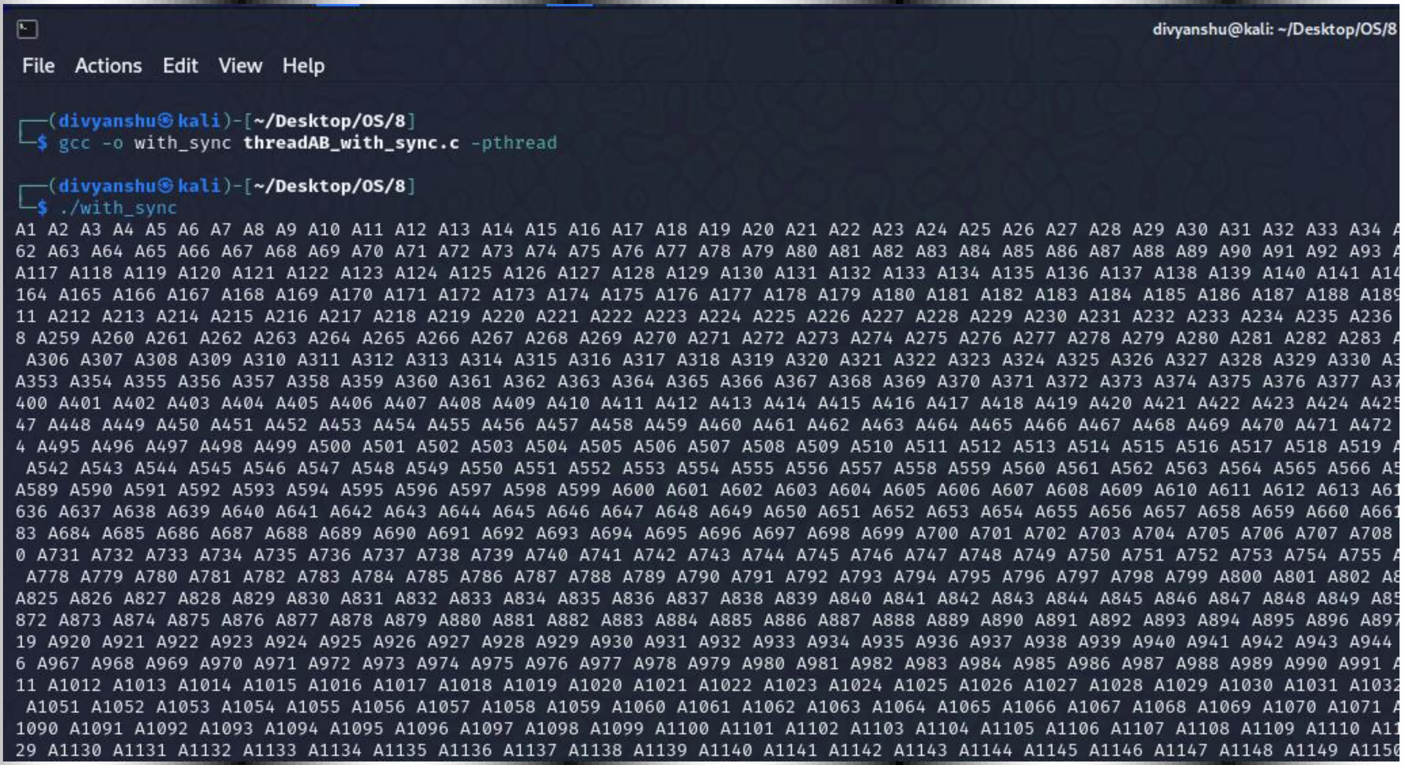
Que 3. Discuss the concurrency of this case study with and without peterson's synchronization.

Two Threads → Thread. A → prints 'A' 100000 times.
→ Thread B → prints number from 1 to 100,000.

	# without Synchronization	# with synchronization
① output	i.e, AA123 A4AAA5678....	i.e. A1 A2 A3 A4 ....
	② → Both thread runs freely resulting unpredictdble output.	② → Only one thread can print at a time (i.e, in its turn)
	③ → This runs faster due to absence of synchronization overhead.	③ → will be slower due to sync. mechanism.
	④ → Inconsistent output	④ → more consistent output



```
┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ gcc -o without_sync threadAB_without_sync.c -pthread

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./without_sync
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
6 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 1
5 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 1
4 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 2
3 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 3
2 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 3
1 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 4
0 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 4
9 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 5
8 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 6
7 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 6
6 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 7
5 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 7
4 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 8
3 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 9
2 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 9
1 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 10
40 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 106
```



```
┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ gcc -o with_sync threadAB_with_sync.c -pthread

┌──(divyanshu㉿kali)-[~/Desktop/OS/8]
└─$ ./with_sync
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28 A29 A30 A31 A32 A33 A34 A
62 A63 A64 A65 A66 A67 A68 A69 A70 A71 A72 A73 A74 A75 A76 A77 A78 A79 A80 A81 A82 A83 A84 A85 A86 A87 A88 A89 A90 A91 A92 A93 A
A117 A118 A119 A120 A121 A122 A123 A124 A125 A126 A127 A128 A129 A130 A131 A132 A133 A134 A135 A136 A137 A138 A139 A140 A141 A14
164 A165 A166 A167 A168 A169 A170 A171 A172 A173 A174 A175 A176 A177 A178 A179 A180 A181 A182 A183 A184 A185 A186 A187 A188 A189
11 A212 A213 A214 A215 A216 A217 A218 A219 A220 A221 A222 A223 A224 A225 A226 A227 A228 A229 A230 A231 A232 A233 A234 A235 A236
8 A259 A260 A261 A262 A263 A264 A265 A266 A267 A268 A269 A270 A271 A272 A273 A274 A275 A276 A277 A278 A279 A280 A281 A282 A283 A
 A306 A307 A308 A309 A310 A311 A312 A313 A314 A315 A316 A317 A318 A319 A320 A321 A322 A323 A324 A325 A326 A327 A328 A329 A330 A3
A353 A354 A355 A356 A357 A358 A359 A360 A361 A362 A363 A364 A365 A366 A367 A368 A369 A370 A371 A372 A373 A374 A375 A376 A377 A37
400 A401 A402 A403 A404 A405 A406 A407 A408 A409 A410 A411 A412 A413 A414 A415 A416 A417 A418 A419 A420 A421 A422 A423 A424 A425
47 A448 A449 A450 A451 A452 A453 A454 A455 A456 A457 A458 A459 A460 A461 A462 A463 A464 A465 A466 A467 A468 A469 A470 A471 A472
4 A495 A496 A497 A498 A499 A500 A501 A502 A503 A504 A505 A506 A507 A508 A509 A510 A511 A512 A513 A514 A515 A516 A517 A518 A519 A
 A542 A543 A544 A545 A546 A547 A548 A549 A550 A551 A552 A553 A554 A555 A556 A557 A558 A559 A560 A561 A562 A563 A564 A565 A566 A5
A589 A590 A591 A592 A593 A594 A595 A596 A597 A598 A599 A600 A601 A602 A603 A604 A605 A606 A607 A608 A609 A610 A611 A612 A613 A61
636 A637 A638 A639 A640 A641 A642 A643 A644 A645 A646 A647 A648 A649 A650 A651 A652 A653 A654 A655 A656 A657 A658 A659 A660 A661
83 A684 A685 A686 A687 A688 A689 A690 A691 A692 A693 A694 A695 A696 A697 A698 A699 A700 A701 A702 A703 A704 A705 A706 A707 A708
0 A731 A732 A733 A734 A735 A736 A737 A738 A739 A740 A741 A742 A743 A744 A745 A746 A747 A748 A749 A750 A751 A752 A753 A754 A755 A
 A778 A779 A780 A781 A782 A783 A784 A785 A786 A787 A788 A789 A790 A791 A792 A793 A794 A795 A796 A797 A798 A799 A800 A801 A802 A8
A825 A826 A827 A828 A829 A830 A831 A832 A833 A834 A835 A836 A837 A838 A839 A840 A841 A842 A843 A844 A845 A846 A847 A848 A849 A85
872 A873 A874 A875 A876 A877 A878 A879 A880 A881 A882 A883 A884 A885 A886 A887 A888 A889 A890 A891 A892 A893 A894 A895 A896 A897
19 A920 A921 A922 A923 A924 A925 A926 A927 A928 A929 A930 A931 A932 A933 A934 A935 A936 A937 A938 A939 A940 A941 A942 A943 A944
6 A967 A968 A969 A970 A971 A972 A973 A974 A975 A976 A977 A978 A979 A980 A981 A982 A983 A984 A985 A986 A987 A988 A989 A990 A991 A
11 A1012 A1013 A1014 A1015 A1016 A1017 A1018 A1019 A1020 A1021 A1022 A1023 A1024 A1025 A1026 A1027 A1028 A1029 A1030 A1031 A1032
 A1051 A1052 A1053 A1054 A1055 A1056 A1057 A1058 A1059 A1060 A1061 A1062 A1063 A1064 A1065 A1066 A1067 A1068 A1069 A1070 A1071 A
1090 A1091 A1092 A1093 A1094 A1095 A1096 A1097 A1098 A1099 A1100 A1101 A1102 A1103 A1104 A1105 A1106 A1107 A1108 A1109 A1110 A11
29 A1130 A1131 A1132 A1133 A1134 A1135 A1136 A1137 A1138 A1139 A1140 A1141 A1142 A1143 A1144 A1145 A1146 A1147 A1148 A1149 A1150
```