[CSS553]
Name: Rahul Ranjan
Roll No: 20CS8016

Assignment 3

1. Write a code to speedup search in a large unsorted integer array by creating (n) child processes each to search the array just to know if the element is found in the array. Each child should search a non-overlapping portion of the array. The parent takes input to the array elements randomly and asks the user to input the element to be searched. The search process should stop as soon as any one of the child finds a success. The child process exits with status code(1) if successful. Use `wait()` system call for parent child synchronization.

2. A program where a parent forks (n) child processes to speedup finding all prime numbers in some given range [x,y], where both x & y are provided as input and both are > 1000. Each child writes the prime numbers found in a common file which the parent should create before creating any of the child process. The parent suspends it till all the child processes completes their execution. Use `wait()` system call for parent child synchronization.

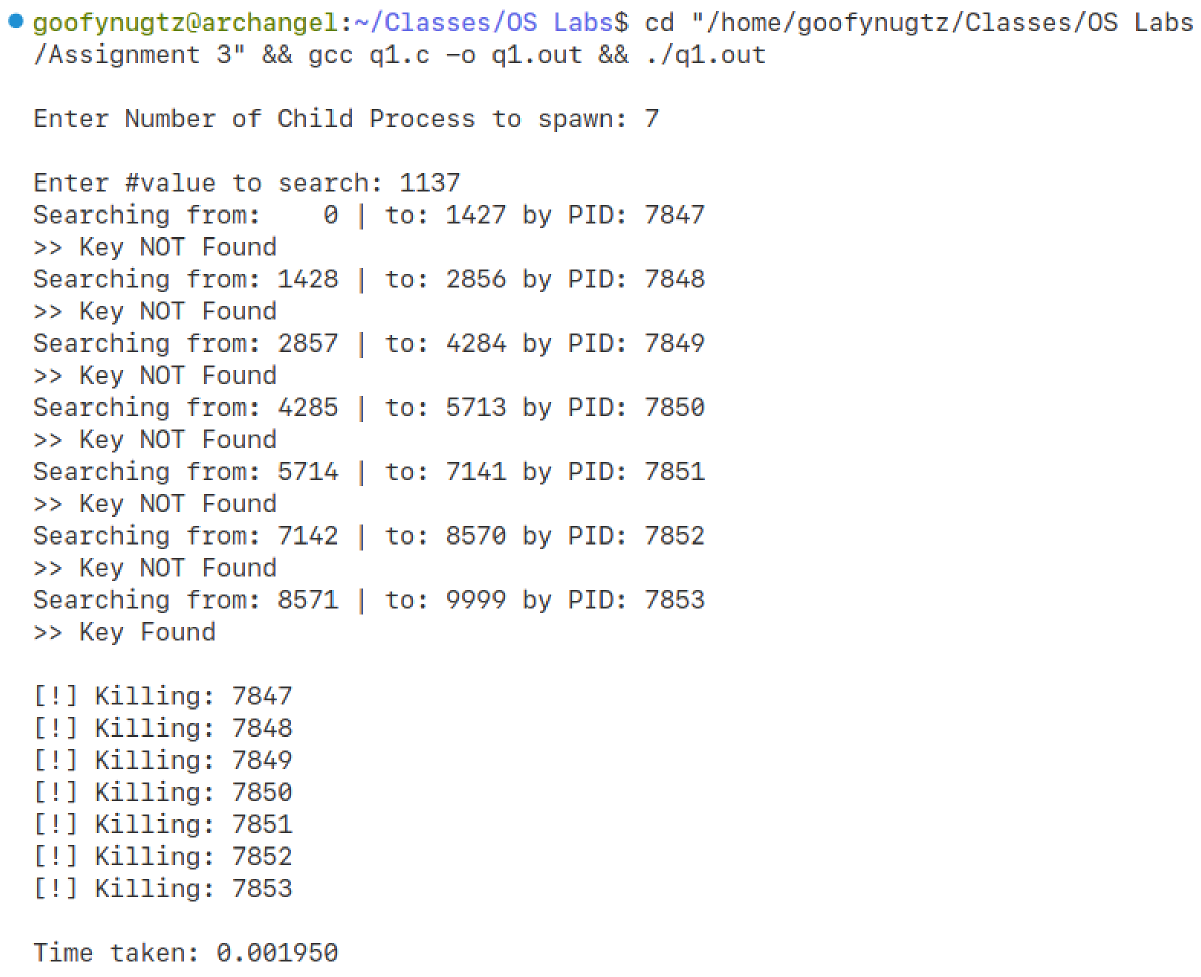
3. Solve Problem (A1) using `waitpid()` system call for parent child synchronization.

4. Solve Problem (A2) using `waitpid()` system call for parent child synchronization.

Answer 1 – Search a value in unsorted array with ‘n’ child process

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#define CC 4

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q3: Search a value in large unsorted array with 'n' child process.
*/

int r;

pid_t c_pids[CC];
int cc = CC; // Child Count

void elementFoundHandler(int x){
  for (int i = 0; i < cc; i++){
    if (c_pids[i] != 0){
      printf("[!] Killing: %d\n", c_pids[i]);
      kill(c_pids[i], SIGINT);
    }
  }
}

void elementNotFoundHandler(int x){}

bool linearSearch(int * arr, int n, int i, int cc, int key){
  int start = (i*n/cc);
  int end = ((i+1)*n/cc)-1;
  printf( "Searching from: M    | to: M    by PID: %d\n", start, end, getpid());
  for (int j = start; j <= end; j++)
    if (arr[j] == key) return true;
  return false;
}

int main(void){
  signal(SIGUSR2, elementNotFoundHandler);
  signal(SIGUSR1, elementFoundHandler);
  int n = 10000;
  int arr[n];
  srand(22);
  for (int i = 0; i < n; i++) arr[i] = rand() % 10000;

  FILE * fp;
  fp = fopen("array.txt", "w");
  for (int i = 0; i < n; i+=10){
    for (int j = 0; j < 10; j++)
      fprintf(fp, "M    ", arr[i+j]);
    fprintf(fp, "\n");
  }
  fclose(fp);
  pid_t wait_p;

  printf("\nEnter Number of Child Process to spawn: ");
  scanf("%d", &cc);

  int status = 0;
  int key; // 1137 // 1410 present in 3/4 divs

  printf("\nEnter #value to search: ");
  scanf("%d", &key);

  clock_t start, end;
  start = clock();

  for (int i = 0; i < cc; i++){
    c_pids[i] = fork();
    if (c_pids[i] == 0){
      bool found = linearSearch(arr, n, i, cc, key);
      if (found){
        printf(">> Key Found\n\n");
        kill(getppid(), SIGUSR1);
        exit(EXIT_SUCCESS);
      } else {
        printf(">> Key NOT Found\n");
        kill(getppid(), SIGUSR2);
        exit(EXIT_FAILURE);
      }
    }
  }
  while((wait_p = wait(&status)) > 0);
  end = clock();
  double time_used = ((double)(end - start))/CLOCKS_PER_SEC;
  printf("\nTime taken: %lf\n\n", time_used);
  return 0;
}
```

>> Output – Case 1



```
● goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
  /Assignment 3" && gcc q1.c -o q1.out && ./q1.out

  Enter Number of Child Process to spawn: 7

  Enter #value to search: 1137
  Searching from:    0 | to: 1427 by PID: 7847
  >> Key NOT Found
  Searching from: 1428 | to: 2856 by PID: 7848
  >> Key NOT Found
  Searching from: 2857 | to: 4284 by PID: 7849
  >> Key NOT Found
  Searching from: 4285 | to: 5713 by PID: 7850
  >> Key NOT Found
  Searching from: 5714 | to: 7141 by PID: 7851
  >> Key NOT Found
  Searching from: 7142 | to: 8570 by PID: 7852
  >> Key NOT Found
  Searching from: 8571 | to: 9999 by PID: 7853
  >> Key Found

  [!] Killing: 7847
  [!] Killing: 7848
  [!] Killing: 7849
  [!] Killing: 7850
  [!] Killing: 7851
  [!] Killing: 7852
  [!] Killing: 7853

  Time taken: 0.001950
```

Answer 2 – Finding all prime numbers in range x,y

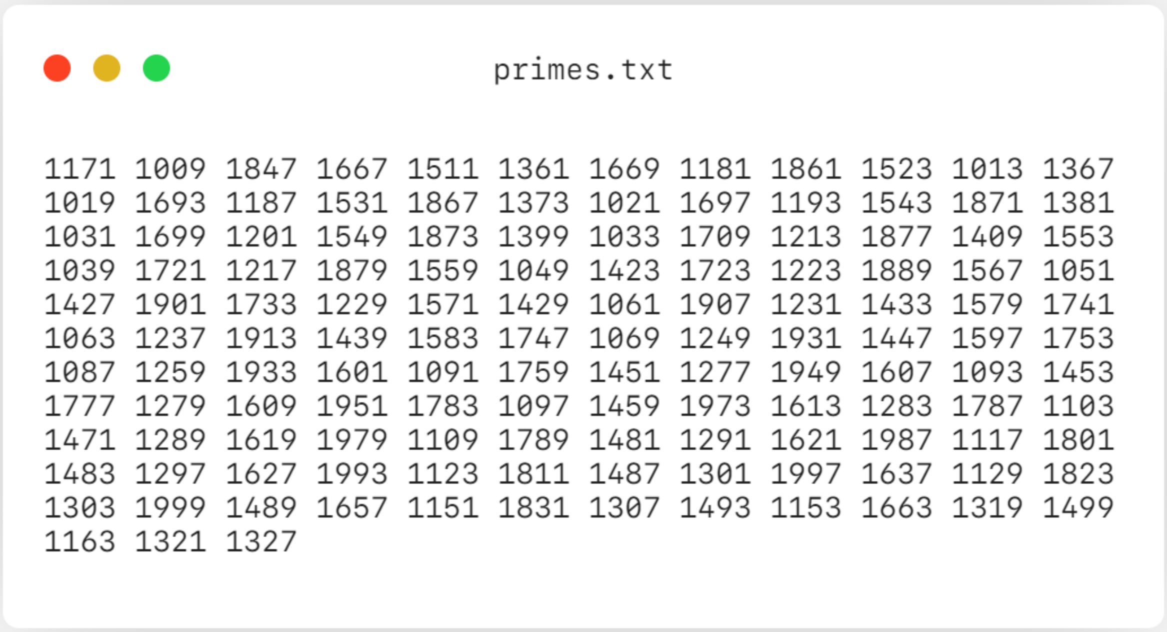
```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdlib.h>
#include <stdbool.h>
#define CC 2

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q2: Find ALL Prime numbers in range [x,y].
*/

pid_t c_pids[CC];
int complete = 0;
int cc = CC;

void kill_child_process(){
  for (int i = 0; i < cc; i++)
    if (c_pids[i] != 0){
      printf("[!] Killing: %d\n", c_pids[i]);
      kill(c_pids[i], SIGINT);
    }
}

void increment_and_check(){
  complete++;
  if (complete == CC) kill_child_process();
}

bool isPrime(int n){
  if (n <= 1)
    return false;
  for (int i = 2; i*i <= n; i++)
    if (n % i == 0) return false;
  return true;
}

void getPrimes(int x, int y){
  for (int i = x; i <= (int)y; i++)
    if(isPrime(i)){
      FILE * log;
      log = fopen("primes.txt", "a");
      fprintf(log, "%d ", i);
      fclose(log);
    }
}
```

```c

int main(void){
  signal(SIGUSR1, increment_and_check);
  signal(SIGUSR2, increment_and_check);
  int x,y, n;
  printf("\nEnter #x #y and #n: ");
  scanf("%d", &x);
  scanf("%d", &y);
  scanf("%d", &n);
  printf("\n");
  cc = n;
  printf("PPID: %d\n", getpid());
  // Resetting the logfile
  FILE * log;
  log = fopen("primes2.txt", "w");
  fclose(log);

  pid_t wait_p;
  int ans, start, end, status;

  for (int i = 0; i < cc; i++){
    start = i*(y-x)/cc + x+1;
    end = (i+1)*(y-x)/cc + x;
    c_pids[i] = fork();
    if (c_pids[i] == 0){
      printf("Start: %d | End: %d PID @ %d\n", start, end, getpid());
      getPrimes(start, end);
      if (i == 0) kill(getppid(), SIGUSR1);
      else if (i == 1) kill(getppid(), SIGUSR2);
      exit(0);
    }
  }
  while ((wait_p = wait(&status)) > 0);
  printf("Exiting from %d\n", getpid());
  return 0;
}
```

>> Output

```
● goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
  /Assignment 3" && gcc q2.c -o q2.out && ./q2.out

  Enter #x #y and #n: 1000 2000 6

  PPID: 7939
  Start: 1001 | End: 1166 PID @ 7986
  Start: 1167 | End: 1333 PID @ 7987
  Start: 1334 | End: 1500 PID @ 7988
  Start: 1501 | End: 1666 PID @ 7989
  Start: 1667 | End: 1833 PID @ 7990
  Start: 1834 | End: 2000 PID @ 7991
  Exiting from 7939
```

>> Output File



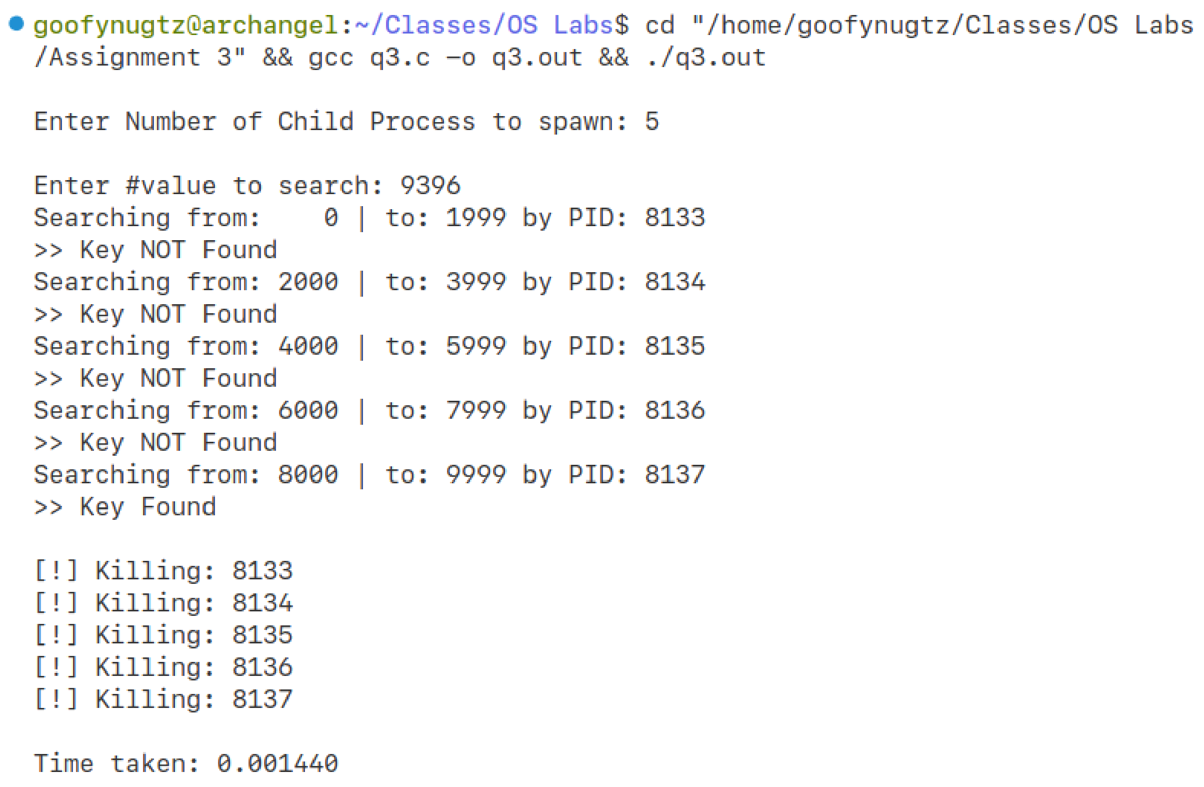
primes.txt

```
1171 1009 1847 1667 1511 1361 1669 1181 1861 1523 1013 1367
1019 1693 1187 1531 1867 1373 1021 1697 1193 1543 1871 1381
1031 1699 1201 1549 1873 1399 1033 1709 1213 1877 1409 1553
1039 1721 1217 1879 1559 1049 1423 1723 1223 1889 1567 1051
1427 1901 1733 1229 1571 1429 1061 1907 1231 1433 1579 1741
1063 1237 1913 1439 1583 1747 1069 1249 1931 1447 1597 1753
1087 1259 1933 1601 1091 1759 1451 1277 1949 1607 1093 1453
1777 1279 1609 1951 1783 1097 1459 1973 1613 1283 1787 1103
1471 1289 1619 1979 1109 1789 1481 1291 1621 1987 1117 1801
1483 1297 1627 1993 1123 1811 1487 1301 1997 1637 1129 1823
1303 1999 1489 1657 1151 1831 1307 1493 1153 1663 1319 1499
1163 1321 1327
```

Answer 3 – Searching a value in array (`waitpid`)

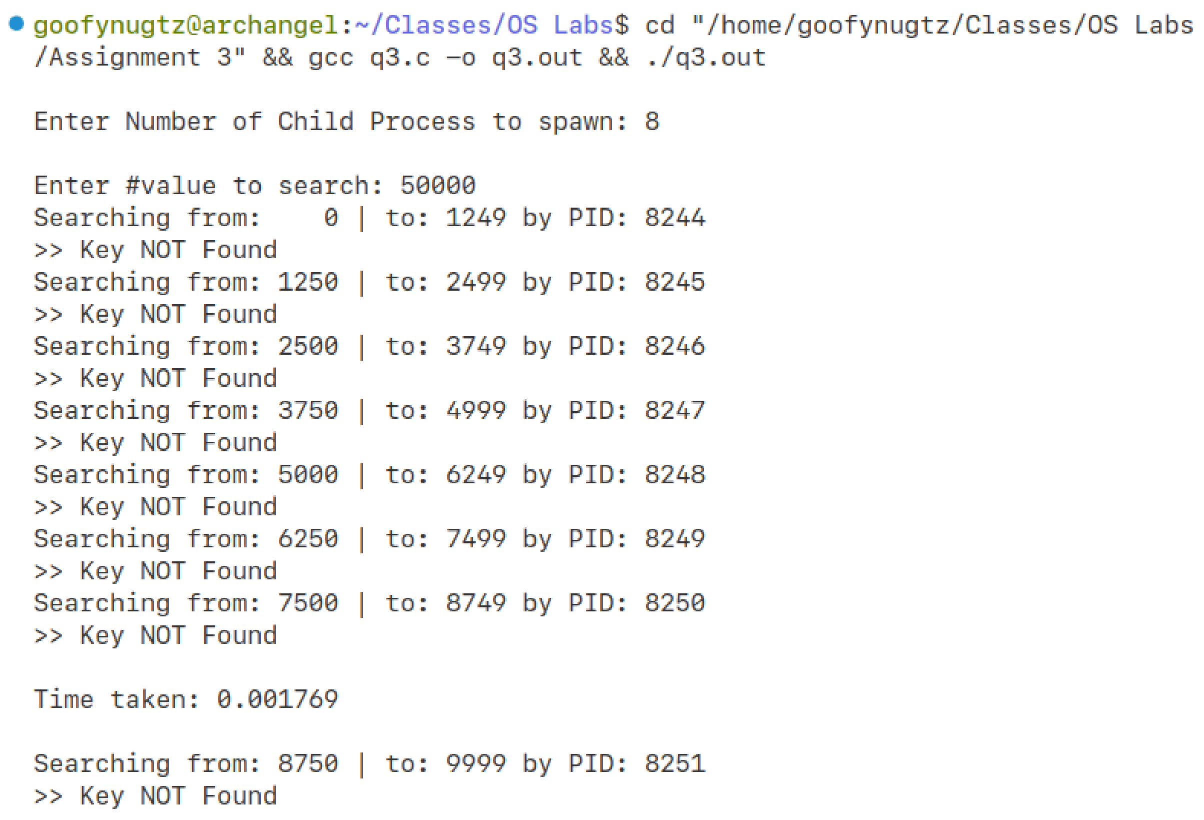
```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#define CC 4

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q3: Search a value in large unsorted array with 'n' child process.
*/

int r;

pid_t c_pids[CC];
int cc = CC; // Child Count

void elementFoundHandler(int x){
  for (int i = 0; i < cc; i++){
    if (c_pids[i] != 0){
      printf("[!] Killing: %d\n", c_pids[i]);
      kill(c_pids[i], SIGINT);
    }
  }
}

void elementNotFoundHandler(int x){}

bool linearSearch(int * arr, int n, int i, int cc, int key){
  int start = (i*n/cc);
  int end = ((i+1)*n/cc)-1;
  printf( "Searching from: M   | to: M   by PID: %d\n", start, end, getpid());
  for (int j = start; j <= end; j++)
    if (arr[j] == key) return true;
  return false;
}

int main(void){
  signal(SIGUSR2, elementNotFoundHandler);
  signal(SIGUSR1, elementFoundHandler);
  int n = 10000;
  int arr[n];
  srand(22);
  for (int i = 0; i < n; i++) arr[i] = rand() % 10000;

  FILE * fp;
  fp = fopen("array.txt", "w");
  for (int i = 0; i < n; i+=10){
    for (int j = 0; j < 10; j++)
      fprintf(fp, "M   ", arr[i+j]);
    fprintf(fp, "\n");
  }
  fclose(fp);
  pid_t wait_p;

  printf("\nEnter Number of Child Process to spawn: ");
  scanf("%d", &cc);

  int status = 0;
  int key; // 1137 // 1410 present in 3/4 divs

  printf("\nEnter #value to search: ");
  scanf("%d", &key);

  clock_t start, end;
  start = clock();

  for (int i = 0; i < cc; i++){
    c_pids[i] = fork();
    if (c_pids[i] == 0){
      bool found = linearSearch(arr, n, i, cc, key);
      if (found){
        printf(">> Key Found\n\n");
        kill(getppid(), SIGUSR1);
        exit(EXIT_SUCCESS);
      } else {
        printf(">> Key NOT Found\n");
        kill(getppid(), SIGUSR2);
        exit(EXIT_FAILURE);
      }
    }
  }
  for (int i = 0; i < CC; i++)
    wait_p = waitpid(c_pids[i], &status, 0);

  end = clock();
  double time_used = ((double)(end - start))/CLOCKS_PER_SEC;
  printf("\nTime taken: %lf\n\n", time_used);
  return 0;
}
```

>> Output



```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 3" && gcc q3.c -o q3.out && ./q3.out

Enter Number of Child Process to spawn: 5

Enter #value to search: 9396
Searching from:    0 | to: 1999 by PID: 8133
>> Key NOT Found
Searching from: 2000 | to: 3999 by PID: 8134
>> Key NOT Found
Searching from: 4000 | to: 5999 by PID: 8135
>> Key NOT Found
Searching from: 6000 | to: 7999 by PID: 8136
>> Key NOT Found
Searching from: 8000 | to: 9999 by PID: 8137
>> Key Found

[!] Killing: 8133
[!] Killing: 8134
[!] Killing: 8135
[!] Killing: 8136
[!] Killing: 8137

Time taken: 0.001440
```

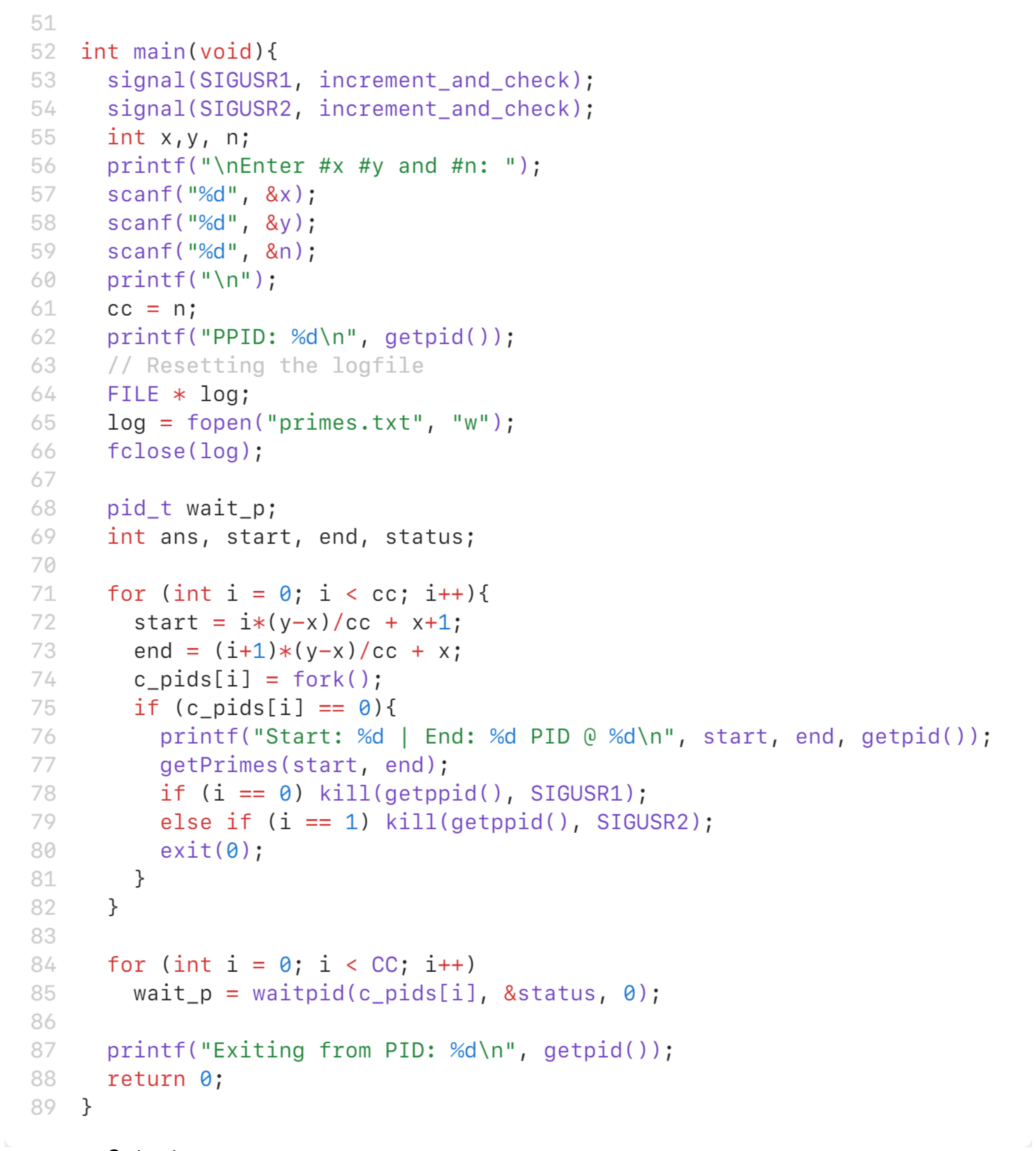
>> Output



```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 3" && gcc q3.c -o q3.out && ./q3.out

Enter Number of Child Process to spawn: 8

Enter #value to search: 50000
Searching from:    0 | to: 1249 by PID: 8244
>> Key NOT Found
Searching from: 1250 | to: 2499 by PID: 8245
>> Key NOT Found
Searching from: 2500 | to: 3749 by PID: 8246
>> Key NOT Found
Searching from: 3750 | to: 4999 by PID: 8247
>> Key NOT Found
Searching from: 5000 | to: 6249 by PID: 8248
>> Key NOT Found
Searching from: 6250 | to: 7499 by PID: 8249
>> Key NOT Found
Searching from: 7500 | to: 8749 by PID: 8250
>> Key NOT Found

Time taken: 0.001769

Searching from: 8750 | to: 9999 by PID: 8251
>> Key NOT Found
```

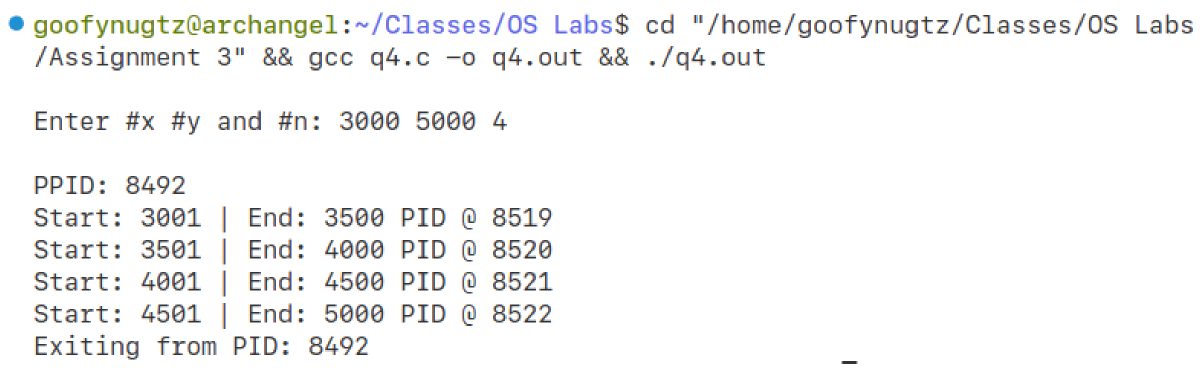
Answer 4 – Finding all prime numbers (`waitpid`)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdlib.h>
#include <stdbool.h>
#define CC 2

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q4: Find ALL Prime numbers in range [x,y].
*/

pid_t c_pids[CC];
int complete = 0;
int cc = CC;

void kill_child_process(){
  for (int i = 0; i < cc; i++)
    if (c_pids[i] != 0){
      printf("[!] Killing: %d\n", c_pids[i]);
      kill(c_pids[i], SIGINT);
    }
}

void increment_and_check(){
  complete++;
  if (complete == CC) kill_child_process();
}

bool isPrime(int n){
  if (n <= 1)
    return false;
  for (int i = 2; i*i <= n; i++)
    if (n % i == 0) return false;
  return true;
}

void getPrimes(int x, int y){
  for (int i = x; i <= (int)y; i++)
    if(isPrime(i)){
      FILE * log;
      log = fopen("primes4.txt", "a");
      fprintf(log, "%d ", i);
      fclose(log);
    }
}
```



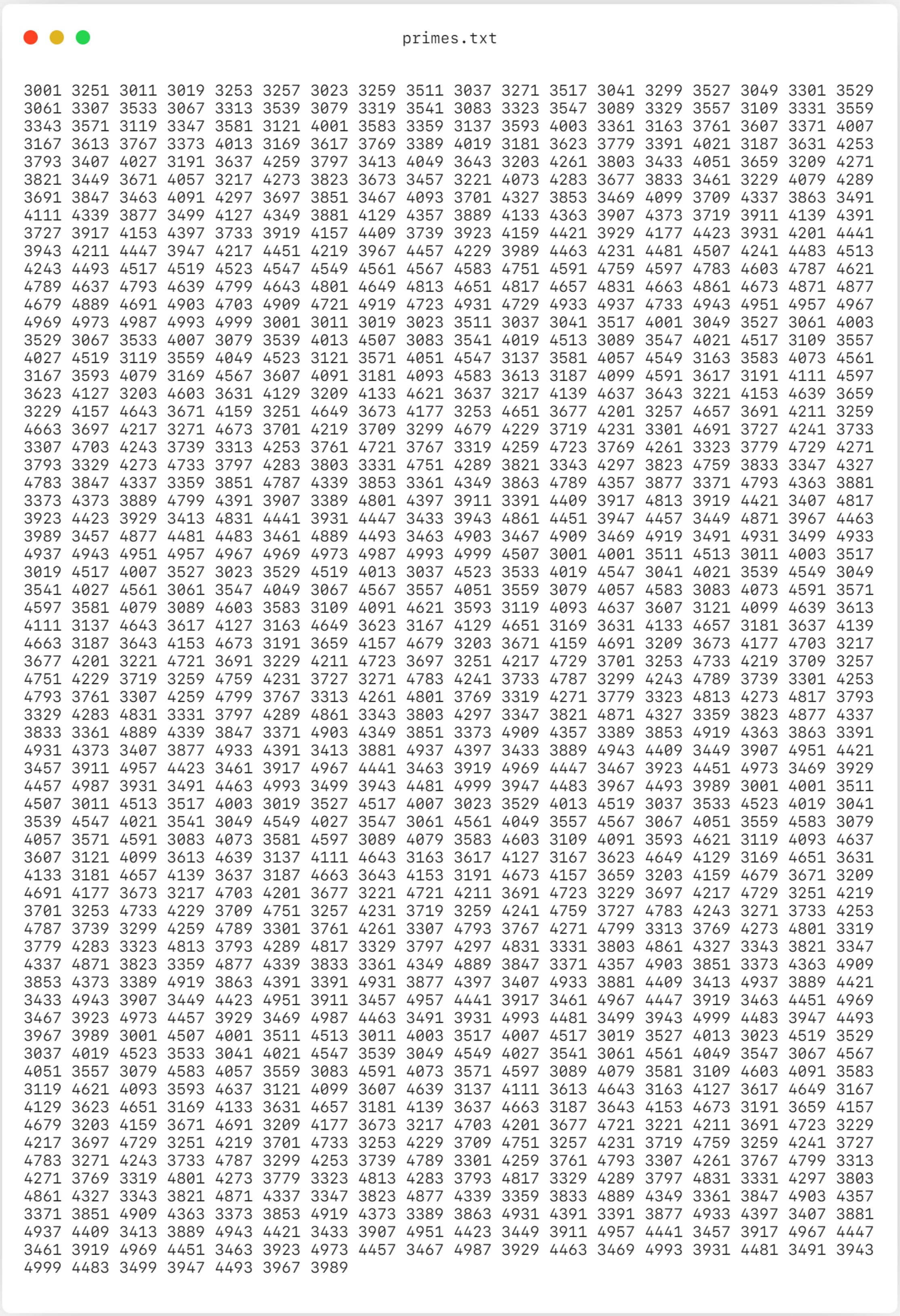
```c

int main(void){
  signal(SIGUSR1, increment_and_check);
  signal(SIGUSR2, increment_and_check);
  int x,y, n;
  printf("\nEnter #x #y and #n: ");
  scanf("%d", &x);
  scanf("%d", &y);
  scanf("%d", &n);
  printf("\n");
  cc = n;
  printf("PPID: %d\n", getpid());
  // Resetting the logfile
  FILE * log;
  log = fopen("primes.txt", "w");
  fclose(log);

  pid_t wait_p;
  int ans, start, end, status;

  for (int i = 0; i < cc; i++){
    start = i*(y-x)/cc + x+1;
    end = (i+1)*(y-x)/cc + x;
    c_pids[i] = fork();
    if (c_pids[i] == 0){
      printf("Start: %d | End: %d PID @ %d\n", start, end, getpid());
      getPrimes(start, end);
      if (i == 0) kill(getppid(), SIGUSR1);
      else if (i == 1) kill(getppid(), SIGUSR2);
      exit(0);
    }
  }

  for (int i = 0; i < CC; i++)
    wait_p = waitpid(c_pids[i], &status, 0);

  printf("Exiting from PID: %d\n", getpid());
  return 0;
}
```

>> Output



```
● goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
  /Assignment 3" && gcc q4.c -o q4.out && ./q4.out

  Enter #x #y and #n: 3000 5000 4

  PPID: 8492
  Start: 3001 | End: 3500 PID @ 8519
  Start: 3501 | End: 4000 PID @ 8520
  Start: 4001 | End: 4500 PID @ 8521
  Start: 4501 | End: 5000 PID @ 8522
  Exiting from PID: 8492
```

>> Output File



primes.txt

```
3001 3251 3011 3019 3253 3257 3023 3259 3511 3037 3271 3517 3041 3299 3527 3049 3301 3529
3061 3307 3533 3067 3313 3539 3079 3319 3541 3083 3323 3547 3089 3329 3557 3109 3331 3559
3343 3571 3119 3347 3581 3121 4001 3583 3359 3137 3593 4003 3361 3163 3761 3607 3371 4007
3167 3613 3767 3373 4013 3169 3617 3769 3389 4019 3181 3623 3779 3391 4021 3187 3631 4253
3793 3407 4027 3191 3637 4259 3797 3413 4049 3643 3203 4261 3803 3433 4051 3659 3209 4271
3821 3449 3671 4057 3217 4273 3823 3673 3457 3221 4073 4283 3677 3833 3461 3229 4079 4289
3691 3847 3463 4091 4297 3697 3851 3467 4093 3701 4327 3853 3469 4099 3709 4337 3863 3491
4111 4339 3877 3499 4127 4349 3881 4129 4357 3889 4133 4363 3907 4373 3719 3911 4139 4391
3727 3917 4153 4397 3733 3919 4157 4409 3739 3923 4159 4421 3929 4177 4423 3931 4201 4441
3943 4211 4447 3947 4217 4451 4219 3967 4457 4229 3989 4463 4231 4481 4507 4241 4483 4513
4243 4493 4517 4519 4523 4547 4549 4561 4567 4583 4751 4591 4759 4597 4783 4603 4787 4621
4789 4637 4793 4639 4799 4643 4801 4649 4813 4651 4817 4657 4831 4663 4861 4673 4871 4877
4679 4889 4691 4903 4703 4909 4721 4919 4723 4931 4729 4933 4937 4733 4943 4951 4957 4967
4969 4973 4987 4993 4999 3001 3011 3019 3023 3511 3037 3041 3517 4001 3049 3527 3061 4003
3529 3067 3533 4007 3079 3539 4013 4507 3083 3541 4019 4513 3089 3547 4021 4517 3109 3557
4027 4519 3119 3559 4049 4523 3121 3571 4051 4547 3137 3581 4057 4549 3163 3583 4073 4561
3167 3593 4079 3169 4567 3607 4091 3181 4093 4583 3613 3187 4099 4591 3617 3191 4111 4597
3623 4127 3203 4603 3631 4129 3209 4133 4621 3637 3217 4139 4637 3643 3221 4153 4639 3659
3229 4157 4643 3671 4159 3251 4649 3673 4177 3253 4651 3677 4201 3257 4657 3691 4211 3259
4663 3697 4217 3271 4673 3701 4219 3709 3299 4679 4229 3719 4231 3301 4691 3727 4241 3733
3307 4703 4243 3739 3313 4253 3761 4721 3767 3319 4259 4723 3769 4261 3323 3779 4729 4271
3793 3329 4273 4733 3797 4283 3803 3331 4751 4289 3821 3343 4297 3823 4759 3833 3347 4327
4783 3847 4337 3359 3851 4787 4339 3853 3361 4349 3863 4789 4357 3877 3371 4793 4363 3881
3373 4373 3889 4799 4391 3907 3389 4801 4397 3911 3391 4409 3917 4813 3919 4421 3407 4817
3923 4423 3929 3413 4831 4441 3931 4447 3433 3943 4861 4451 3947 4457 3449 4871 3967 4463
3989 3457 4877 4481 4483 3461 4889 4493 3463 4903 3467 4909 3469 4919 3491 4931 3499 4933
4937 4943 4951 4957 4967 4969 4973 4987 4993 4999 4507 3001 4001 3511 4513 3011 4003 3517
3019 4517 4007 3527 3023 3529 4519 4013 3037 4523 3533 4019 4547 3041 4021 3539 4549 3049
3541 4027 4561 3061 3547 4049 3067 4567 3557 4051 3559 3079 4057 4583 3083 4073 4591 3571
4597 3581 4079 3089 4603 3583 3109 4091 4621 3593 3119 4093 4637 3607 3121 4099 4639 3613
4111 3137 4643 3617 4127 3163 4649 3623 3167 4129 4651 3169 3631 4133 4657 3181 3637 4139
4663 3187 3643 4153 4673 3191 3659 4157 4679 3203 3671 4159 4691 3209 3673 4177 4703 3217
3677 4201 3221 4721 3691 3229 4211 4723 3697 3251 4217 4729 3701 3253 4733 4219 3709 3257
4751 4229 3719 3259 4759 4231 3727 3271 4783 4241 3733 4787 3299 4243 4789 3739 3301 4253
4793 3761 3307 4259 4799 3767 3313 4261 4801 3769 3319 4271 3779 3323 4813 4273 4817 3793
3329 4283 4831 3331 3797 4289 4861 3343 3803 4297 3347 3821 4871 4327 3359 3823 4877 4337
3833 3361 4889 4339 3847 3371 4903 4349 3851 3373 4909 4357 3389 3853 4919 4363 3863 3391
4931 4373 3407 3877 4933 4391 3413 3881 4937 4397 3433 3889 4943 4409 3449 3907 4951 4421
3457 3911 4957 4423 3461 3917 4967 4441 3463 3919 4969 4447 3467 3923 4451 4973 3469 3929
4457 4987 3931 3491 4463 4993 3499 3943 4481 4999 3947 4483 3967 4493 3989 3001 4001 3511
4507 3011 4513 3517 4003 3019 3527 4517 4007 3023 3529 4013 4519 3037 3533 4523 4019 3041
3539 4547 4021 3541 3049 4549 4027 3547 3061 4561 4049 3557 4567 3067 4051 3559 4583 3079
4057 3571 4591 3083 4073 3581 4597 3089 4079 3583 4603 3109 4091 3593 4621 3119 4093 4637
3607 3121 4099 3613 4639 3137 4111 4643 3163 3617 4127 3167 3623 4649 4129 3169 4651 3631
4133 3181 4657 4139 3637 3187 4663 3643 4153 3191 4673 4157 3659 3203 4159 4679 3671 3209
4691 4177 3673 3217 4703 4201 3677 3221 4721 4211 3691 4723 3229 3697 4217 4729 3251 4219
3701 3253 4733 4229 3709 4751 3257 4231 3719 3259 4241 4759 3727 4783 4243 3271 3733 4253
4787 3739 3299 4259 4789 3301 3761 4261 3307 4793 3767 4271 4799 3313 3769 4273 4801 3319
3779 4283 3323 4813 3793 4289 4817 3329 3797 4297 4831 3331 3803 4861 4327 3343 3821 3347
4337 4871 3823 3359 4877 4339 3833 3361 4349 4889 3847 3371 4357 4903 3851 3373 4363 4909
3853 4373 3389 4919 3863 4391 3391 4931 3877 4397 3407 4933 3881 4409 3413 4937 3889 4421
3433 4943 3907 3449 4423 4951 3911 3457 4957 4441 3917 3461 4967 4447 3919 3463 4451 4969
3467 3923 4973 4457 3929 3469 4987 4463 3491 3931 4993 4481 3499 3943 4999 4483 3947 4493
3967 3989 3001 4507 4001 3511 4513 3011 4003 3517 4007 4517 3019 3527 4013 3023 4519 3529
3037 4019 4523 3533 3041 4021 4547 3539 3049 4549 4027 3541 3061 4561 4049 3547 3067 4567
4051 3557 3079 4583 4057 3559 3083 4591 4073 3571 4597 3089 4079 3581 3109 4603 4091 3583
3119 4621 4093 3593 4637 3121 4099 3607 4639 3137 4111 3613 4643 3163 4127 3617 4649 3167
4129 3623 4651 3169 4133 3631 4657 3181 4139 3637 4663 3187 3643 4153 4673 3191 3659 4157
4679 3203 4159 3671 4691 3209 4177 3673 3217 4703 4201 3677 4721 3221 4211 3691 4723 3229
4217 3697 4729 3251 4219 3701 4733 3253 4229 3709 4751 3257 4231 3719 4759 3259 4241 3727
4783 3271 4243 3733 4787 3299 4253 3739 4789 3301 4259 3761 4793 3307 4261 3767 4799 3313
4271 3769 3319 4801 4273 3779 3323 4813 4283 3793 4817 3329 4289 3797 4831 3331 4297 3803
4861 4327 3343 3821 4871 4337 3347 3823 4877 4339 3359 3833 4889 4349 3361 3847 4903 4357
3371 3851 4909 4363 3373 3853 4919 4373 3389 3863 4931 4391 3391 3877 4933 4397 3407 3881
4937 4409 3413 3889 4943 4421 3433 3907 4951 4423 3449 3911 4957 4441 3457 3917 4967 4447
3461 3919 4969 4451 3463 3923 4973 4457 3467 4987 3929 4463 3469 4993 3931 4481 3491 3943
4999 4483 3499 3947 4493 3967 3989
```

All compiler codes are uploaded here