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

Assignment 2

1. Write a code such that it takes input n and then the parent process p creates a child c1, then c1 creates c2, c2 creates c3 ... and so on till cn. Each process including the parent should display the pid & ppid once in the terminal and also store the values in a common file (log.txt). The parent can create the file log.txt and all its child processes can get access it. Use suitable signals such that if Ctrl+C is pressed make sure that all the child processes gets terminated but the parent must keep running.

2. Write a code to speedup search in a large unsorted integer array by creating 2 child processes each to search half of the array. The search process should stop as soon as one of the child finds a success. Follow these steps to solve it.

3. Modify the solution in Assignment A2 such that it implements a generalized version of n process instead of 2 i.e. the parent process should also accept an input 'n' and create 'n' number of processes for achieve more parallelism. Try increasing the value of n and find if the amount of speedup achieved is observable (use difference of system clock time before & after the computation)

4. A program where a parent forks two child processes to speedup finding ONE prime number in some given range [x,y], where both x & y are provided as input and both are > 10000. Follow the steps to solve it

5. A program where a parent forks two child processes to speedup finding ALL prime numbers in some given range [x,y]

6. Design and implement a suitable signaling protocol where a parent forks two child processes to speedup finding 'p' number of prime numbers in some given range [x,y]. Input p, x & y. The prime numbers may be stored in prime.txt file by the child processes.

Answer 1: Ctrl+C terminates all child but the parent.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q1: Child spawns another child process.
Crtl + C kills all child process but not the parent.
*/

void ignore_signal(int x){
  printf("\t[Alert!] Recieved SIG:%d. Signal Ignored by PID: %d\n", x, getpid());
  return;
}

void terminate_process(int x){
  printf("\t[Alert!] Recieved SIG:%d. Terminating PID: %d\n", x, getpid());
  signal(SIGINT, SIG_DFL);
  kill(getpid(), SIGINT);
}

void main(void){
  pid_t c_pid, wait_p;
  int status = 0;

  int n = 5;
  printf(">> PID: %d\n", getpid());
  signal(SIGINT, ignore_signal);

  for (int i = 0; i < n; i++){
    c_pid = fork();
    if (c_pid == 0){
      signal(SIGINT, terminate_process);
      printf(">> PID: %d PPID: %d\n", getpid(), getppid());
    } else break;
  }
  while((wait_p = wait(&status)) > 0);
  while(1);
}

```

>> Output

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q1.c -o q1.out && ./q1.out
>> PID: 6367
>> PID: 6368 PPID: 6367
>> PID: 6369 PPID: 6368
>> PID: 6370 PPID: 6369
>> PID: 6371 PPID: 6370
>> PID: 6372 PPID: 6371
^C      [Alert!] Recieved SIG:2. Terminating PID: 6372
        [Alert!] Recieved SIG:2. Signal Ignored by PID: 6367
        [Alert!] Recieved SIG:2. Terminating PID: 6368
        [Alert!] Recieved SIG:2. Terminating PID: 6371
        [Alert!] Recieved SIG:2. Terminating PID: 6370
        [Alert!] Recieved SIG:2. Terminating PID: 6369
^C      [Alert!] Recieved SIG:2. Signal Ignored by PID: 6367
^C      [Alert!] Recieved SIG:2. Signal Ignored by PID: 6367
```

```
goofynugtz@archangel: ~
goofynugtz@archangel:~$ ps -ef | grep q1
goofynu+    6367    6344  0 19:57 pts/1    00:00:00 ./q1.out
goofynu+    6368    6367  0 19:57 pts/1    00:00:00 ./q1.out
goofynu+    6369    6368  0 19:57 pts/1    00:00:00 ./q1.out
goofynu+    6370    6369  0 19:57 pts/1    00:00:00 ./q1.out
goofynu+    6371    6370  0 19:57 pts/1    00:00:00 ./q1.out
goofynu+    6372    6371 99 19:57 pts/1    00:00:06 ./q1.out
goofynu+    6420    6334  0 19:57 pts/0    00:00:00 grep --color=auto q1
goofynugtz@archangel:~$ ps -ef | grep q1
goofynu+    6367    6344 51 19:57 pts/1    00:00:13 ./q1.out
goofynu+    6500    6334  0 19:57 pts/0    00:00:00 grep --color=auto q1
goofynugtz@archangel:~$
```

Answer 2: Search a value in unsorted array with 2 child process.

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

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q2: Searching a value in a large unsorted array with '2' child process.
*/

int r;

pid_t c_pids[CC];

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

void elementFoundHandler(int x){
  kill_child_process();
}

void elementNotFoundHandler(int x){}

bool linearSearch(int * arr, int n, int i, int key){
  int start = (i*n/2);
  int end = ((i+1)*n/2)-1;
  printf("Searching in arr[ %d ] .. arr[ %d ] by PID: %d\n", start, end, getpid());
  for (int j = start; j <= end; j++)
    if (arr[j] == key) return 1;
  return 0;
}

int main(void){
  signal(35, elementFoundHandler);
  signal(36, elementNotFoundHandler);

  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){
```

```c
    for (int j = 0; j < 10; j++)
      fprintf(fp, "M   ", arr[i+j]);
    fprintf(fp, "\n");
  }
  fclose(fp);

  int key; // 1137 9396

  printf("\nEnter #value to search: ");
  scanf("%d", &key);
  for (int i = 0; i < 2; i++){
    c_pids[i] = fork();
    if (c_pids[i] == 0){
      if (linearSearch(arr, n, i, key)){
        printf(">> Key Found\n\n");
        kill(getppid(), 35);
        exit(EXIT_SUCCESS);
      }
      else {
        printf(">> Key NOT found\n\n");
        kill(getppid(), 36);
        exit(EXIT_FAILURE);
      }
    }
    if (c_pids[i] != 0){
      r = i;
      printf(">> [r] is set to %d\n", r);
    }
  }
  return 0;
}
```

>> Output – Case 1: Key present in array

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q2.c -o q2.out && ./q2.out

Enter #value to search: 1137
>> [r] is set to 0
Searching in arr[ 0 ] .. arr[ 4999 ] by PID: 6651
>> Key NOT found

>> [r] is set to 1
Searching in arr[ 5000 ] .. arr[ 9999 ] by PID: 6652
>> Key Found
```

>> Output – Case 2: Key **not** present in array

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q2.c -o q2.out && ./q2.out

Enter #value to search: 40000
>> [r] is set to 0
Searching in arr[ 0 ] .. arr[ 4999 ] by PID: 6740
>> [r] is set to 1
>> Key NOT found

Searching in arr[ 5000 ] .. arr[ 9999 ] by PID: 6741
>> Key NOT found
```

Answer 3: 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;
```

```c

  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 2" && gcc q3.c -o q3.out && ./q3.out

Enter Number of Child Process to spawn: 6

Enter #value to search: 1137
Searching from:    0 | to: 1665 by PID: 6858
>> Key NOT Found
Searching from: 1666 | to: 3332 by PID: 6859
>> Key NOT Found
Searching from: 3333 | to: 4999 by PID: 6860
>> Key NOT Found
Searching from: 5000 | to: 6665 by PID: 6861
>> Key NOT Found
Searching from: 6666 | to: 8332 by PID: 6862
>> Key NOT Found
Searching from: 8333 | to: 9999 by PID: 6863
>> Key Found

[!] Killing: 6858
[!] Killing: 6859
[!] Killing: 6860
[!] Killing: 6861
[!] Killing: 6862
[!] Killing: 6863

Time taken: 0.001642
```

```
/* The array can be
regenerated by having
seed=22 and % = 10000;

(ref: line 48-50) */
```

>> Output – Case 2

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q3.c -o q3.out && ./q3.out

Enter Number of Child Process to spawn: 4

Enter #value to search: 50000
Searching from:    0 | to: 2499 by PID: 7091
>> Key NOT Found
Searching from: 2500 | to: 4999 by PID: 7092
>> Key NOT Found
Searching from: 5000 | to: 7499 by PID: 7093
>> Key NOT Found
Searching from: 7500 | to: 9999 by PID: 7094
>> Key NOT Found

Time taken: 0.000839
```

>> Output – Case 3

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q3.c -o q3.out && ./q3.out

Enter Number of Child Process to spawn: 4

Enter #value to search: 1410
Searching from:    0 | to: 2499 by PID: 6970
>> Key Found

Searching from: 2500 | to: 4999 by PID: 6971
[!] Killing: 6970
[!] Killing: 6971
>> Key Found
Searching from: 5000 | to: 7499 by PID: 6972
>> Key NOT Found
Searching from: 7500 | to: 9999 by PID: 6973
>> Key Found

[!] Killing: 6970
[!] Killing: 6971
[!] Killing: 6972
[!] Killing: 6973

Time taken: 0.001176
```

Answer 4: Find ONE prime number 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

Q4: Find ONE Prime number in range [x,y].
*/

pid_t c_pids[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);
    }
}

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;
}

bool get_first_prime(int x, int y, int * ans){
  printf("Start: %d | End: %d by PID: %d\n", x, y, getpid());
  for (int i = x; i <= (int)y; i++)
    if (isPrime(i)){
      (*ans) = i;
      return true;
    }
  return false;
}

int main(void){
  signal(SIGUSR1, kill_child_process);
  int x,y;
  printf("\nEnter #x and #y: ");
  scanf("%d", &x);
  scanf("%d", &y);
```

```c

  int ans, start, end, cc = CC;
  bool found = 0;

  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){
      found = get_first_prime(start, end, &ans);
      if (found){
        printf(">> Got %d.\n\n", ans);
        kill(getppid(), SIGUSR1);
        exit(EXIT_SUCCESS);
      } else exit(EXIT_FAILURE);
    }
    sleep(0.0001);
  }
  return 0;
}
```

>> Output – Case 1

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q4.c -o q4.out && ./q4.out

Enter #x and #y: 1000 2000
Start: 1001 | End: 1500 by PID: 7213
>> Got 1009.

[!] Killing: 7213
[!] Killing: 7214
```

>> Output – Case 2

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q4.c -o q4.out && ./q4.out

Enter #x and #y: 50000 100000
Start: 50001 | End: 75000 by PID: 7316
>> Got 50021.

[!] Killing: 7316
[!] Killing: 7317
```

>> Output – Case 3: No primes in the range

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q4.c -o q4.out && ./q4.out

Enter #x and #y: 10000 10006
Start: 10001 | End: 10003 by PID: 7664
Start: 10004 | End: 10006 by PID: 7665
```

Answer 5: Find 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

Q5: Find ALL Prime numbers in range [x,y].
*/

pid_t c_pids[CC];
int complete = 0;

void kill_child_process(){
  for (int i = 0; i < CC; i++)
    if (c_pids[i] != 0){
      printf("[!] Killing: %d\n", getpid());
      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;
  printf("\nEnter #x and #y: ");
  scanf("%d", &x);
  scanf("%d", &y);
  printf("\n");

  // Resetting the logfile
  FILE * log;
  log = fopen("primes.txt", "w");
  fclose(log);

  int ans, start, end, cc = CC;

  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){
      getPrimes(start, end);
      if (i == 0) kill(getppid(), SIGUSR1);
      else if (i == 1) kill(getppid(), SIGUSR2);
      exit(EXIT_SUCCESS);
    }
  }
  return 0;
}
```

>> Input Case

```
10000 20000
```

>> Output

prime.txt

```
10007 10009 15013 10037 10039 15017 10061 15031 10067 10069 15053 10079 15061 10091
10093 15073 10099 15077 10103 15083 10111 15091 10133 10139 15101 10141 10151 15107
10159 15121 10163 15131 10169 10177 15137 10181 15139 10193 10211 15149 10223 10243
15161 10247 15173 10253 15187 10259 10267 15193 10271 15199 10273 15217 10289 15227
10301 15233 10303 15241 10313 10321 15259 10331 15263 10333 15269 10337 15271 10343
15277 10357 10369 15287 10391 15289 10399 10427 15299 10429 15307 10433 10453 15313
10457 15319 10459 15329 10463 15331 10477 10487 15349 10499 10501 15359 10513 15361
10529 10531 15373 10559 15377 10567 15383 10589 15391 10597 10601 15401 10607 10613
15413 10627 15427 10631 10639 15439 10651 10657 15443 10663 15451 10667 10687 15461
10691 10709 15467 10711 15473 10723 10729 15493 10733 10739 15497 10753 15511 10771
10781 15527 10789 10799 15541 10831 15551 10837 10847 15559 10853 10859 15569 10861
15581 10867 10883 15583 10889 10891 15601 10903 15607 10909 10937 15619 10939 10949
15629 10957 15641 10973 10979 15643 10987 10993 15647 11003 15649 11027 11047 15661
11057 15667 11059 11069 15671 11071 11083 15679 11087 15683 11093 11113 15727 11117
15731 11119 11131 15733 11149 11159 15737 11161 15739 11171 15749 11173 11177 15761
11197 15767 11213 11239 15773 11243 15787 11251 15791 11257 15797 11261 15803 11273
11279 15809 11287 11299 15817 11311 15823 11317 11321 15859 11329 11351 15877 11353
11369 15881 11383 15887 11393 11399 15889 11411 15901 11423 11437 15907 11443 15913
11447 15919 11467 11471 15923 11483 11489 15937 11491 11497 15959 11503 15971 11519
15973 11527 11549 15991 11551 11579 16001 11587 16007 11593 11597 16033 11617 11621
16057 11633 16061 11657 11677 16063 11681 16067 11689 11699 16069 11701 11717 16073
11719 16087 11731 11743 16091 11777 16097 11779 11783 16103 11789 11801 16111 11807
11813 16127 11821 16139 11827 11831 16141 11833 11839 16183 11863 16187 11867 11887
16189 11897 16193 11903 11909 16217 11923 11927 16223 11933 16229 11939 11941 16231
11953 11959 16249 11969 16253 11971 11981 16267 11987 12007 16273 12011 16301 12037
12041 16319 12043 12049 16333 12071 16339 12073 12097 16349 12101 12107 16361 12109
16363 12113 12119 16369 12143 12149 16381 12157 16411 12161 12163 16417 12197 16421
12203 12211 16427 12227 16433 12239 12241 16447 12251 16451 12253 12263 16453 12269
12277 16477 12281 16481 12289 12301 16487 12323 16493 12329 12343 16519 12347 16529
12373 12377 16547 12379 12391 16553 12401 12409 16561 12413 16567 12421 12433 16573
12437 12451 16603 12457 16607 12473 12479 16619 12487 12491 16631 12497 16633 12503
12511 16649 12517 12527 16651 12539 16657 12541 12547 16661 12553 12569 16673 12577
16691 12583 12589 16693 12601 12611 16699 12613 16703 12619 12637 16729 12641 12647
16741 12653 16747 12659 12671 16759 12689 16763 12697 12703 16787 12713 12721 16811
12739 12743 16823 12757 16829 12763 12781 16831 12791 12799 16843 12809 16871 12821
12823 16879 12829 16883 12841 12853 16889 12889 16901 12893 12899 16903 12907 12911
16921 12917 16927 12919 12923 16931 12941 12953 16937 12959 16943 12967 12973 16963
12979 12983 16979 13001 16981 13003 13007 16987 13009 13033 16993 13037 17011 13043
13049 17021 13063 17027 13093 13099 17029 13103 13109 17033 13121 17041 13127 13147
17047 13151 13159 17053 13163 17077 13171 13177 17093 13183 17099 13187 13217 17107
13219 13229 17117 13241 17123 13249 13259 17137 13267 17159 13291 13297 17167 13309
13313 17183 13327 17189 13331 13337 17191 13339 13367 17203 13381 17207 13397 13399
17209 13411 17231 13417 17239 13421 17257 13441 13451 17291 13457 17293 13463 13469
17299 13477 13487 17317 13499 17321 13513 13523 17327 13537 17333 13553 13567 17341
13577 17351 13591 13597 17359 13613 13619 17377 13627 17383 13633 13649 17387 13669
13679 17389 13681 17393 13687 13691 17401 13693 13697 17417 13709 17419 13711 13721
17431 13723 13729 17443 13751 17449 13757 13759 17467 13763 13781 17471 13789 17477
13799 13807 17483 13829 17489 13831 13841 17491 13859 17497 13873 13877 17509 13879
13883 17519 13901 17539 13903 13907 17551 13913 13921 17569 13931 17573 13933 17579
13963 13967 17581 13997 13999 17597 14009 17599 14011 14029 17609 14033 14051 17623
14057 17627 14071 14081 17657 14083 14087 17659 14107 17669 14143 14149 17681 14153
17683 14159 17707 14173 14177 17713 14197 17729 14207 17737 14221 17747 14243 17749
14249 17761 14251 17783 14281 14293 17789 14303 17791 14321 17807 14323 17827 14327
17837 14341 17839 14347 17851 14369 17863 14387 14389 17881 14401 17891 14407 17903
14411 17909 14419 17911 14423 17921 14431 17923 14437 17929 14447 17939 14449 17957
14461 17959 14479 17971 14489 14503 17977 14519 17981 14533 14537 17987 14543 14549
17989 14551 18013 14557 14561 18041 14563 14591 18043 14593 18047 14621 14627 18049
14629 14633 18059 14639 18061 14653 14657 18077 14669 14683 18089 14699 18097 14713
14717 18119 14723 14731 18121 14737 18127 14741 14747 18131 14753 18133 14759 14767
18143 14771 14779 18149 14783 18169 14797 14813 18181 14821 14827 18191 14831 18199
14843 14851 18211 14867 14869 18217 14879 18223 14887 14891 18229 14897 18233 14923
14929 18251 14939 14947 18253 14951 18257 14957 14969 18269 14983 18287 18289 18301
18307 18311 18313 18329 18341 18353 18367 18371 18379 18397 18401 18413 18427 18433
18439 18443 18451 18457 18461 18481 18493 18503 18517 18521 18523 18539 18541 18553
18583 18587 18593 18617 18637 18661 18671 18679 18691 18701 18713 18719 18731 18743
18749 18757 18773 18787 18793 18797 18803 18839 18859 18869 18899 18911 18913 18917
18919 18947 18959 18973 18979 19001 19009 19013 19031 19037 19051 19069 19073 19079
19081 19087 19121 19139 19141 19157 19163 19181 19183 19207 19211 19213 19219 19231
19237 19249 19259 19267 19273 19289 19301 19309 19319 19333 19373 19379 19381 19387
19391 19403 19417 19421 19423 19427 19429 19433 19441 19447 19457 19463 19469 19471
19477 19483 19489 19501 19507 19531 19541 19543 19553 19559 19571 19577 19583 19597
19603 19609 19661 19681 19687 19697 19699 19709 19717 19727 19739 19751 19753 19759
19763 19777 19793 19801 19813 19819 19841 19843 19853 19861 19867 19889 19891 19913
19919 19927 19937 19949 19961 19963 19973 19979 19991 19993 19997
```

Answer 6: Find ‘p’ 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 6

/*
Name: Rahul Ranjan
Roll No: 20CS8016

Q6: Find 'p' Prime numbers in range [x,y].
*/

pid_t c_pids[CC];
int prime_count;
char FILE_NAME[] = "prime.txt";
int SIG = 35;

void kill_child_process(){
  for (int i = 0; i < CC; i++)
    if (c_pids[i] != 0){
      kill(c_pids[i], SIGKILL);
    }
}

void updateCount(int x){
  if (prime_count <= 0){
    kill_child_process();
  } else prime_count--;
}

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){
  printf("Fetching from %d | to: %d by PID: %d\n",x,y, getpid());
  for (int i = x; i <= (int)y; i++)
    if(isPrime(i)){
      FILE * log;
      log = fopen(FILE_NAME, "a");
      kill(getppid(), SIG);
      fprintf(log, "%d\n", i);
      fflush(log);
      fclose(log);
    }
}
```

```c

// Parent Processes for 'p' primes and discards other values.
void cleanup(int p){
  int primes[p];
  FILE * reader = fopen(FILE_NAME, "r+");
  for (int i = 0; i < p; i++) fscanf(reader, "%d\n", &primes[i]);
  fclose(reader);
  FILE * writer = fopen(FILE_NAME, "w");
  for (int i = 0; i < p; i++) fprintf(writer, "%d\n", primes[i]);
}


int main(void){
  signal(SIG, updateCount);

  int x,y,p;
  printf("\nEnter #x #y #p: ");
  scanf("%d", &x);
  scanf("%d", &y);
  scanf("%d", &p);
  printf("\n");
  prime_count = p;
  FILE * log;
  log = fopen(FILE_NAME, "w");
  fclose(log);

  pid_t wait_c; int status = 0;
  int ans, start, end, cc = CC;

  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){
      getPrimes(start, end);
      exit(EXIT_SUCCESS);
    }
  }
  while((wait_c = wait(&status)) > 0);
  cleanup(p);
  return 0;
}
```

>> Input

```
goofynugtz@archangel:~/Classes/OS Labs$ cd "/home/goofynugtz/Classes/OS Labs
/Assignment 2" && gcc q6.c -o q6.out && ./q6.out

Enter #x #y #p: 10000 15000 19

Fetching from 10001 | to: 10833 by PID: 7942
Fetching from 10834 | to: 11666 by PID: 7943
Fetching from 11667 | to: 12500 by PID: 7944
Fetching from 12501 | to: 13333 by PID: 7945
Fetching from 13334 | to: 14166 by PID: 7946
Fetching from 14167 | to: 15000 by PID: 7947
```

>> Output

prime.txt

```
10007
10009
10037
10837
10039
11677
10847
10061
11681
10853
10067
11689
10859
10069
11699
10861
10079
11701
10867
```

All compiler codes are uploaded here.