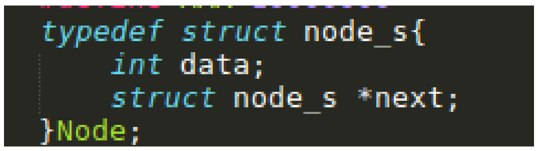
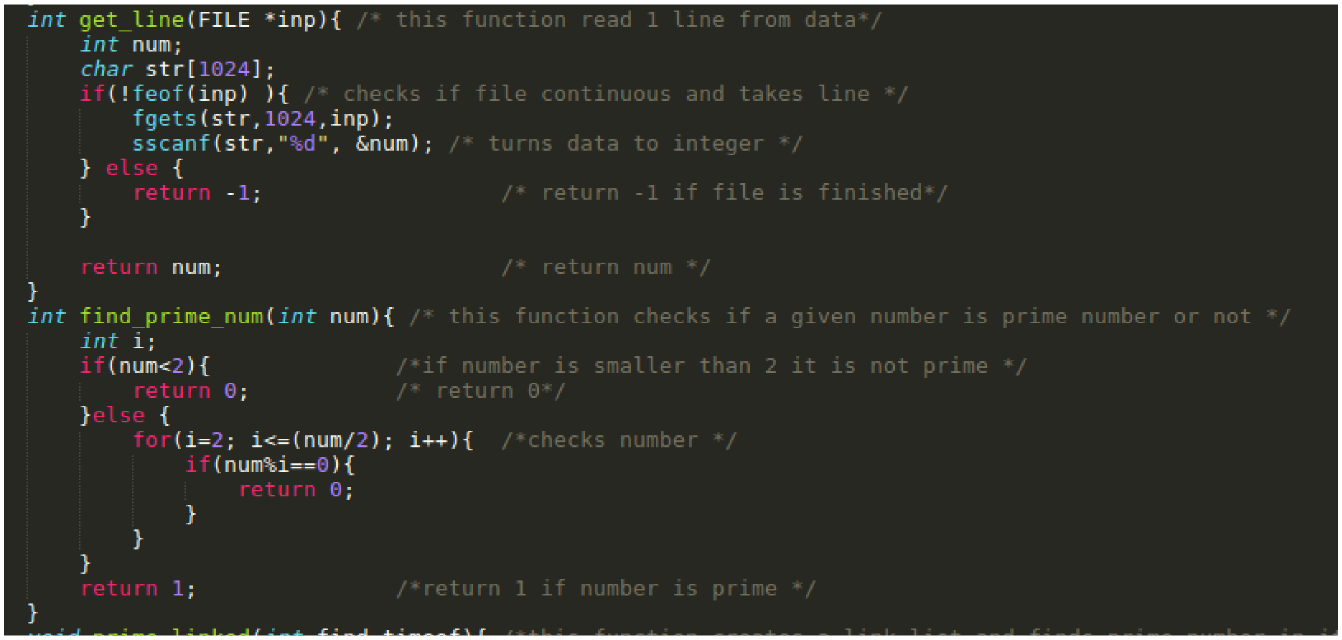
HW10 REPORT

In this homework we had a txt file filled with numbers from 1 to 1000000. We read those numbers from files line by line and created array and linked list and filled them with those numbers. Then find prime numbers and printed them. And calculated running time of those functions and writed to a file those times.

First I made a struct for linked list.



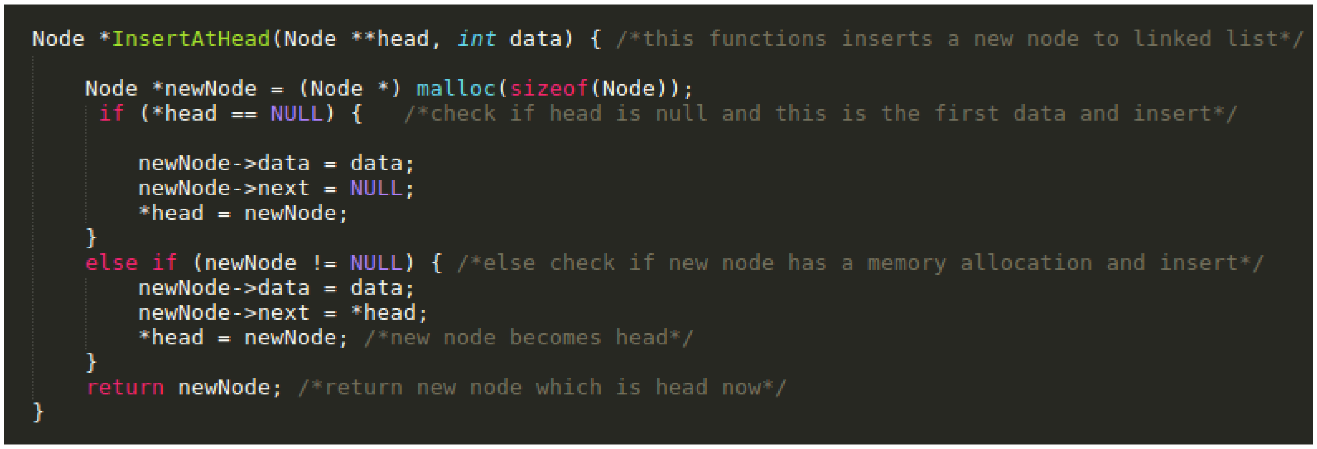
```c
typedef struct node_s{
    int data;
    struct node_s *next;
}Node;
```



```c
int get_line(FILE *inp){ /* this function read 1 line from data*/
    int num;
    char str[1024];
    if(!feof(inp) ){ /* checks if file continuous and takes line */
        fgets(str,1024,inp);
        sscanf(str,"%d", &num); /* turns data to integer */
    } else {
        return -1;                 /* return -1 if file is finished*/
    }

    return num;                    /* return num */
}
int find_prime_num(int num){ /* this function checks if a given number is prime number or not */
    int i;
    if(num<2){                 /*if number is smaller than 2 it is not prime */
        return 0;              /* return 0*/
    }else {
        for(i=2; i<=(num/2); i++){  /*checks number */
            if(num%i==0){
                return 0;
            }
        }
    }
    return 1;                  /*return 1 if number is prime */
}
```

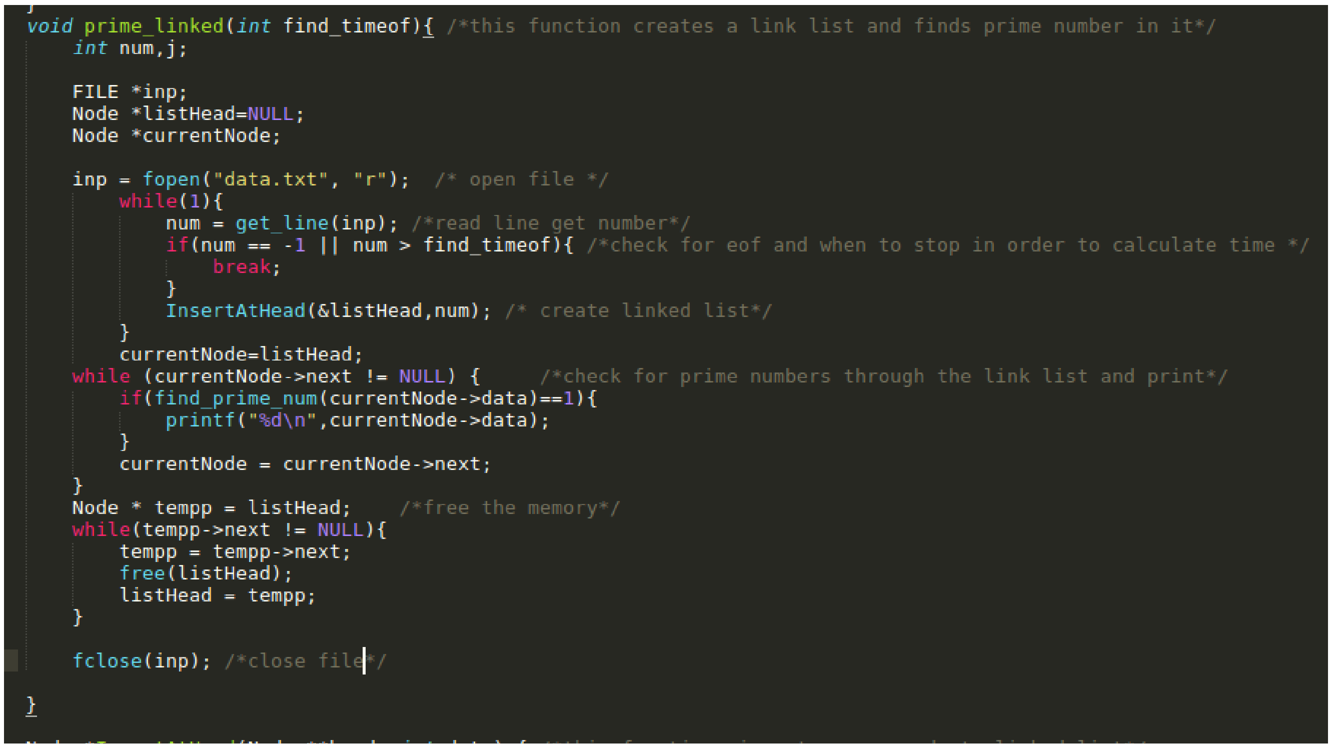
get_line function reads a line from the file, turns it to an integer and returns that integer number. If it is eof function returns -1. It takes file pointer as parameter.

find_prime_num finds if a given number is a prime number or not.



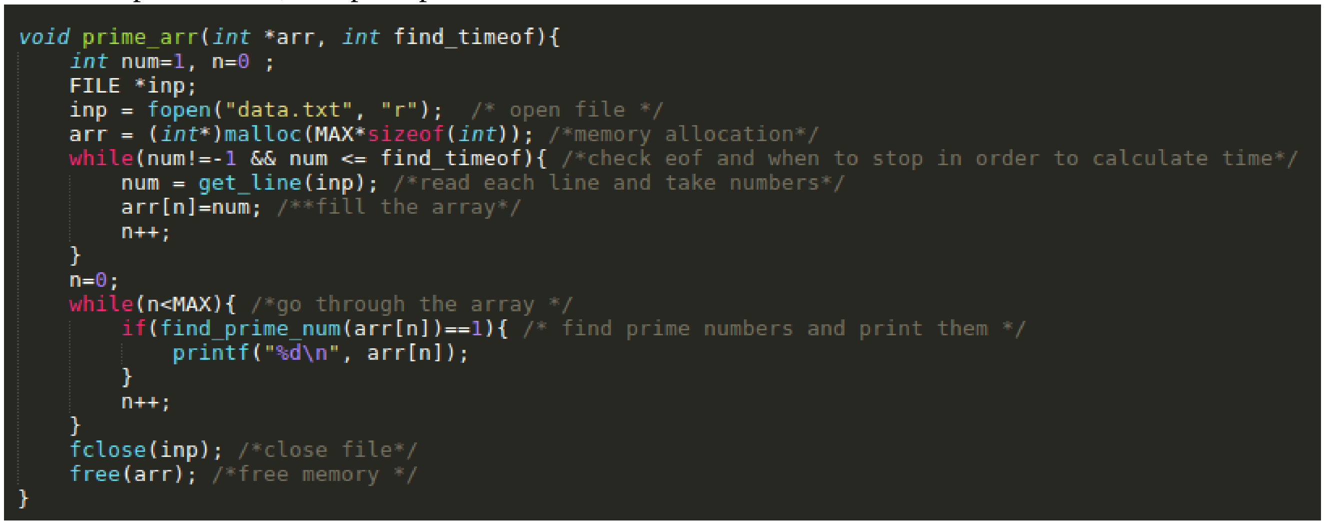
```c
Node *InsertAtHead(Node **head, int data) { /*this functions inserts a new node to linked list*/

    Node *newNode = (Node *) malloc(sizeof(Node));
     if (*head == NULL) {   /*check if head is null and this is the first data and insert*/

        newNode->data = data;
        newNode->next = NULL;
        *head = newNode;
    }
    else if (newNode != NULL) { /*else check if new node has a memory allocation and insert*/
        newNode->data = data;
        newNode->next = *head;
        *head = newNode; /*new node becomes head*/
    }
    return newNode; /*return new node which is head now*/
}
```

InsertAtHead function adds a new node to linked list and the new node becomes head.



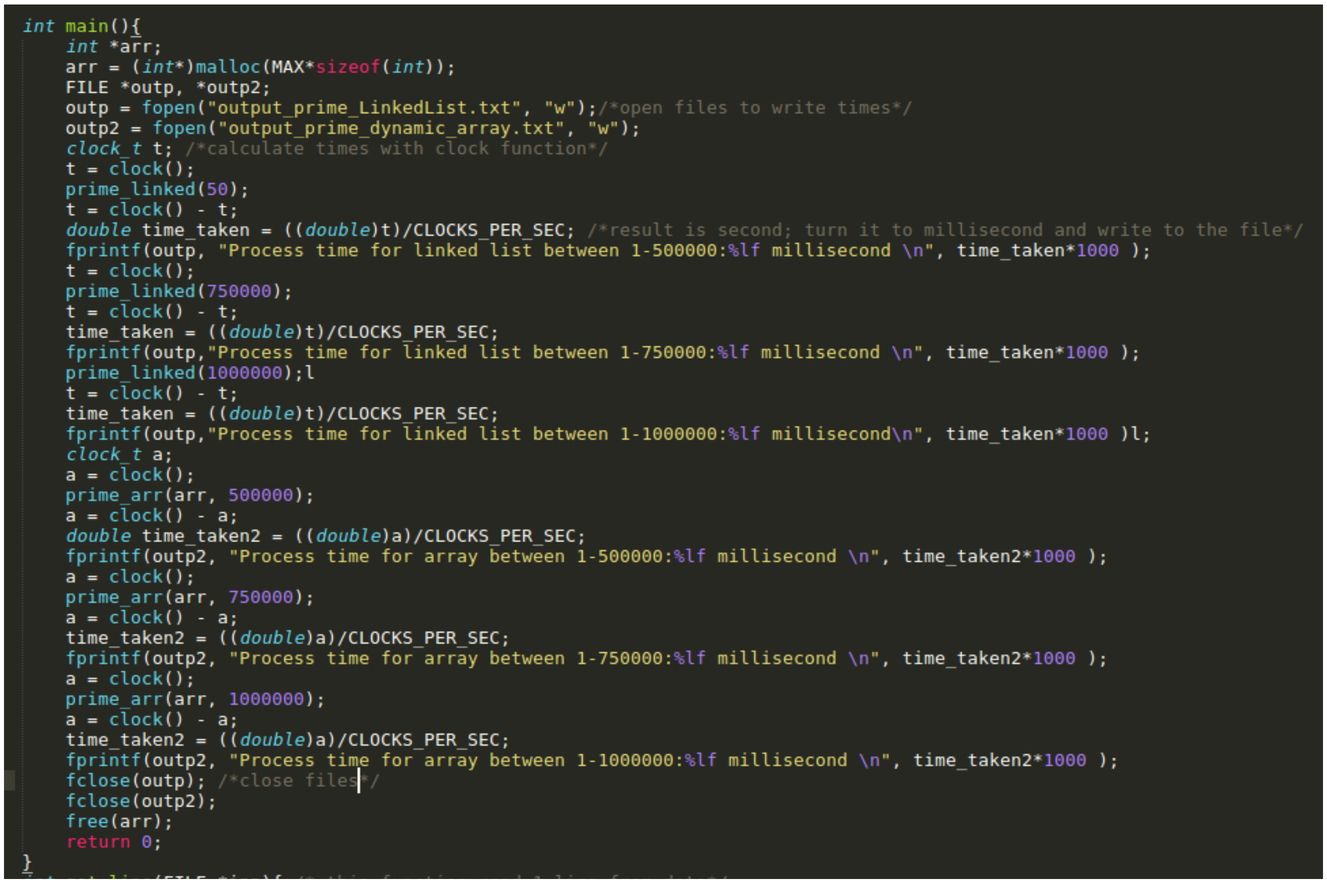
```c
void prime_linked(int find_timeof){ /*this function creates a link list and finds prime number in it*/
    int num,j;

    FILE *inp;
    Node *listHead=NULL;
    Node *currentNode;

    inp = fopen("data.txt", "r");  /* open file */
        while(1){
            num = get_line(inp); /*read line get number*/
            if(num == -1 || num > find_timeof){ /*check for eof and when to stop in order to calculate time */
                break;
            }
            InsertAtHead(&listHead,num); /* create linked list*/
        }
        currentNode=listHead;
    while (currentNode->next != NULL) {     /*check for prime numbers through the link list and print*/
        if(find_prime_num(currentNode->data)==1){
            printf("%d\n",currentNode->data);
        }
        currentNode = currentNode->next;
    }
    Node * tempp = listHead;     /*free the memory*/
    while(tempp->next != NULL){
        tempp = tempp->next;
        free(listHead);
        listHead = tempp;
    }

    fclose(inp); /*close file*/

}
```

In this function we use other funtions we have writed before. Read number from file with get_line, create a linked list and add those numbers to it with InsertAtHead, then check if numbers are prime with find_prime_num, and print prime numbers.



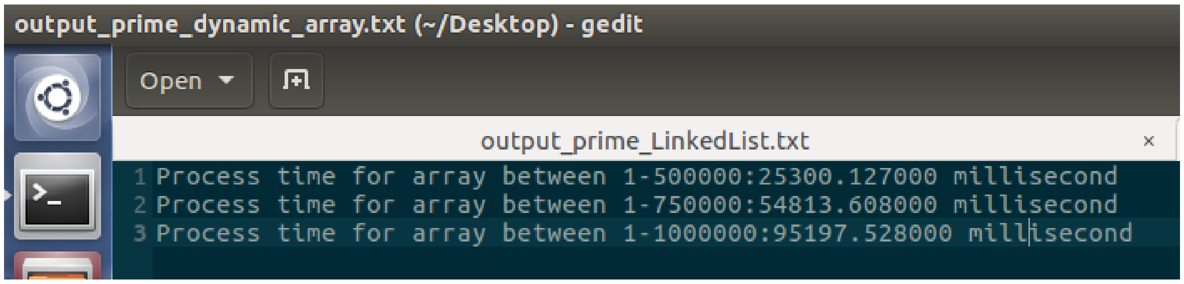
```c
void prime_arr(int *arr, int find_timeof){
    int num=1, n=0 ;
    FILE *inp;
    inp = fopen("data.txt", "r");  /* open file */
    arr = (int*)malloc(MAX*sizeof(int)); /*memory allocation*/
    while(num!=-1 && num <= find_timeof){ /*check eof and when to stop in order to calculate time*/
        num = get_line(inp); /*read each line and take numbers*/
        arr[n]=num; /**fill the array*/
        n++;
    }
    n=0;
    while(n<MAX){ /*go through the array */
        if(find_prime_num(arr[n])==1){ /* find prime numbers and print them */
            printf("%d\n", arr[n]);
        }
        n++;
    }
    fclose(inp); /*close file*/
    free(arr); /*free memory */
}
```

This functions creates an dynamic array. Read number from file with get_line, add those numbers to array, then check if numbers are prime with find_prime_num, and print prime numbers.



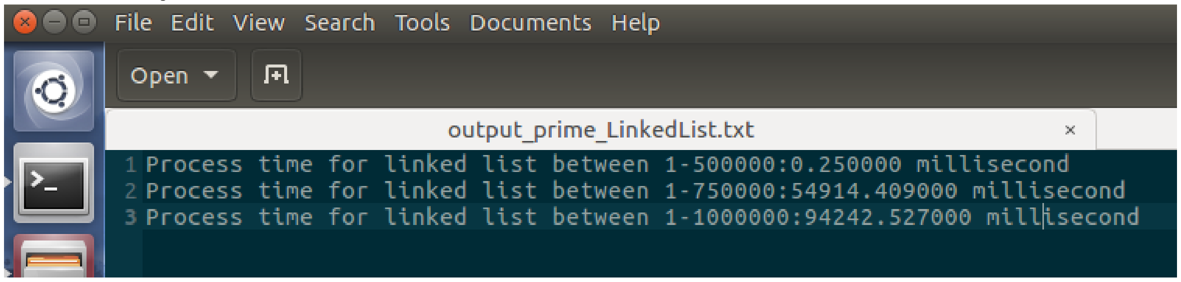
```c
int main(){
    int *arr;
    arr = (int*)malloc(MAX*sizeof(int));
    FILE *outp, *outp2;
    outp = fopen("output_prime_LinkedList.txt", "w");/*open files to write times*/
    outp2 = fopen("output_prime_dynamic_array.txt", "w");
    clock_t t; /*calculate times with clock function*/
    t = clock();
    prime_linked(50);
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; /*result is second; turn it to millisecond and write to the file*/
    fprintf(outp, "Process time for linked list between 1-500000:%lf millisecond \n", time_taken*1000 );
    t = clock();
    prime_linked(750000);
    t = clock() - t;
    time_taken = ((double)t)/CLOCKS_PER_SEC;
    fprintf(outp,"Process time for linked list between 1-750000:%lf millisecond \n", time_taken*1000 );
    prime_linked(1000000);l
    t = clock() - t;
    time_taken = ((double)t)/CLOCKS_PER_SEC;
    fprintf(outp,"Process time for linked list between 1-1000000:%lf millisecond\n", time_taken*1000 )l;
    clock_t a;
    a = clock();
    prime_arr(arr, 500000);
    a = clock() - a;
    double time_taken2 = ((double)a)/CLOCKS_PER_SEC;
    fprintf(outp2, "Process time for array between 1-500000:%lf millisecond \n", time_taken2*1000 );
    a = clock();
    prime_arr(arr, 750000);
    a = clock() - a;
    time_taken2 = ((double)a)/CLOCKS_PER_SEC;
    fprintf(outp2, "Process time for array between 1-750000:%lf millisecond \n", time_taken2*1000 );
    a = clock();
    prime_arr(arr, 1000000);
    a = clock() - a;
    time_taken2 = ((double)a)/CLOCKS_PER_SEC;
    fprintf(outp2, "Process time for array between 1-1000000:%lf millisecond \n", time_taken2*1000 );
    fclose(outp); /*close files*/
    fclose(outp2);
    free(arr);
    return 0;
}
```

In main we calculate times of functions and read those times to txt files.


output_prime_dynamic_array.txt (~/Desktop) - gedit
Open
output_prime_LinkedList.txt

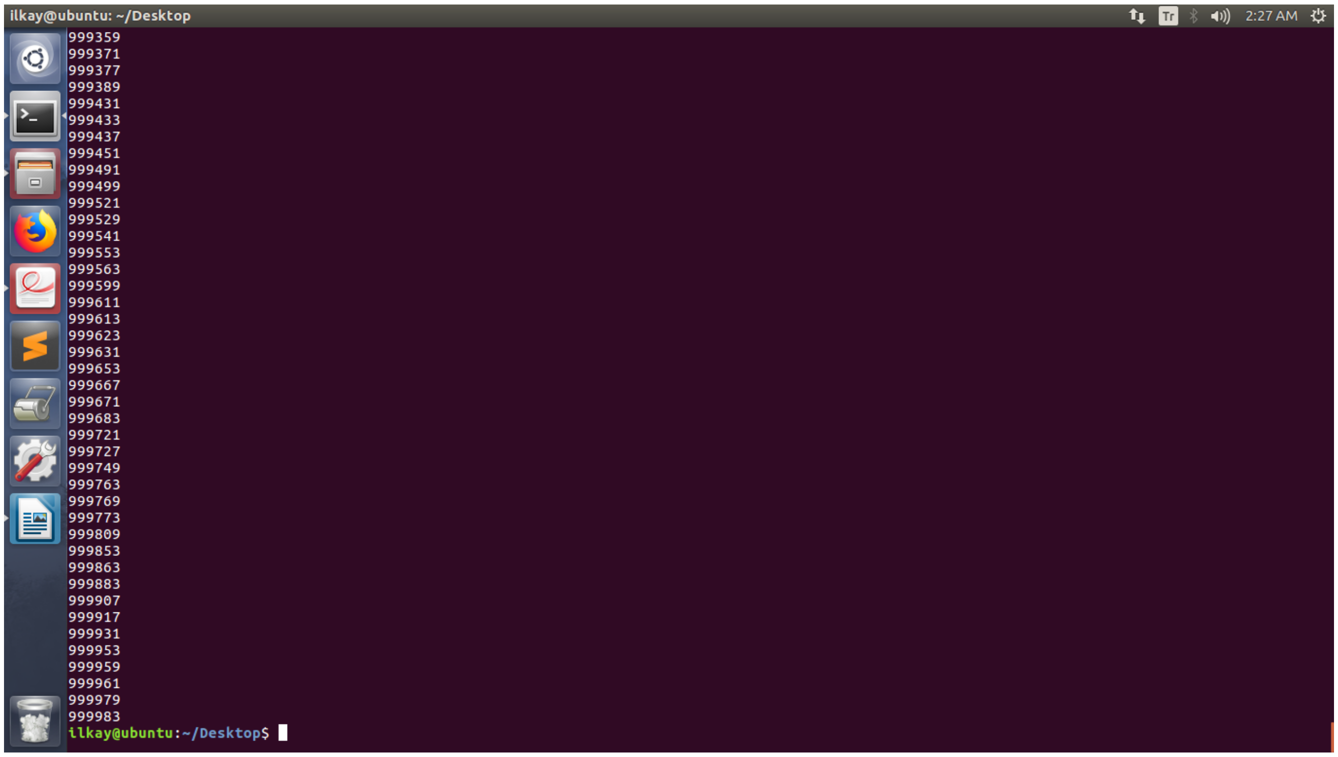
```
Process time for array between 1-500000:25300.127000 millisecond
Process time for array between 1-750000:54813.608000 millisecond
Process time for array between 1-1000000:95197.528000 millisecond
```

Time of array.


File Edit View Search Tools Documents Help
Open
output_prime_LinkedList.txt

```
Process time for linked list between 1-500000:0.250000 millisecond
Process time for linked list between 1-750000:54914.409000 millisecond
Process time for linked list between 1-1000000:94242.527000 millisecond
```

Time of linked list.



```
999359
999371
999377
999389
999431
999433
999437
999451
999491
999499
999521
999529
999541
999553
999563
999599
999611
999613
999623
999631
999653
999667
999671
999683
999721
999727
999749
999763
999769
999773
999809
999853
999863
999883
999907
999917
999931
999953
999959
999961
999979
999983
ilkay@ubuntu:~/Desktop$
```

Program writes prime numbers to terminal like this.