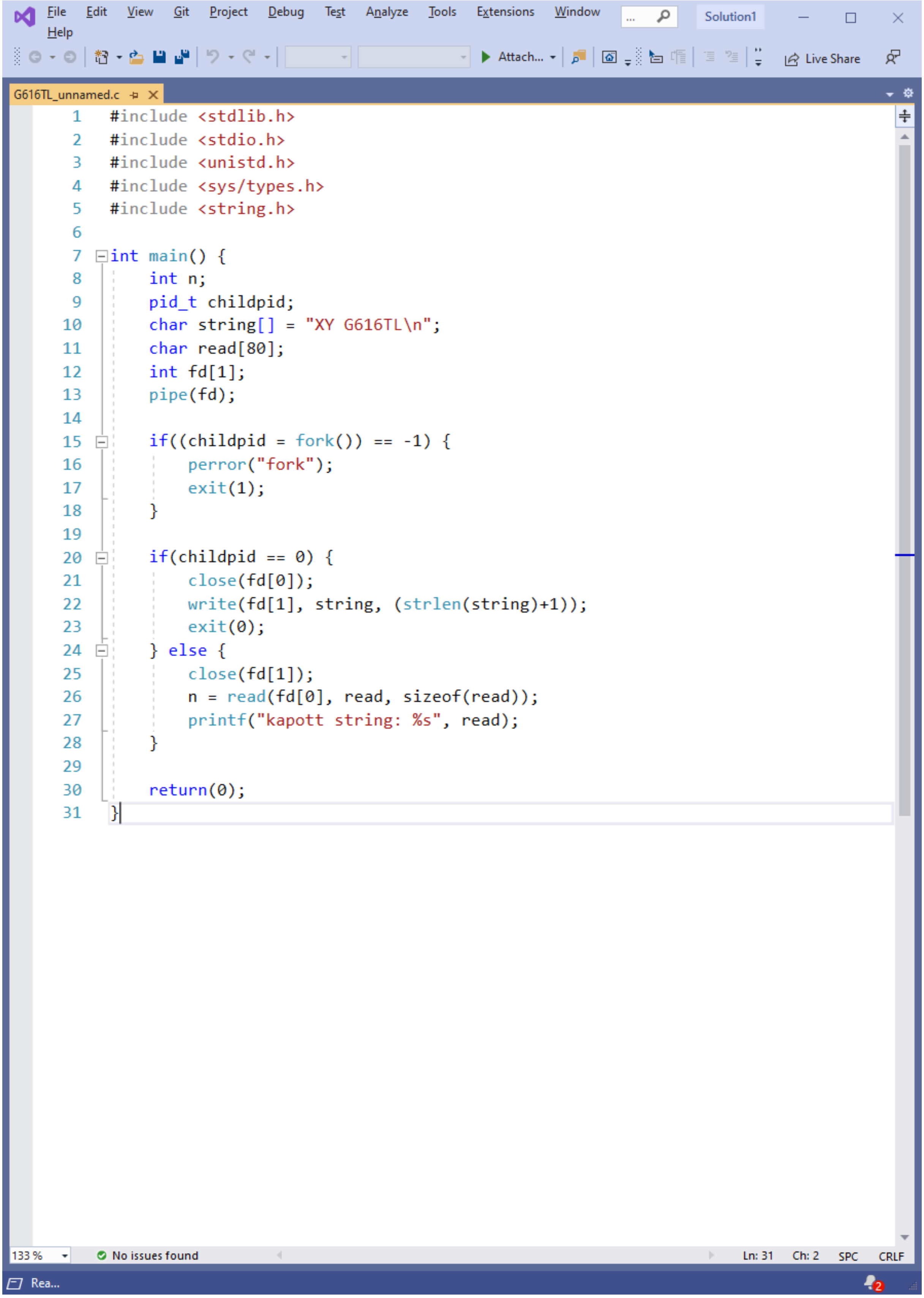
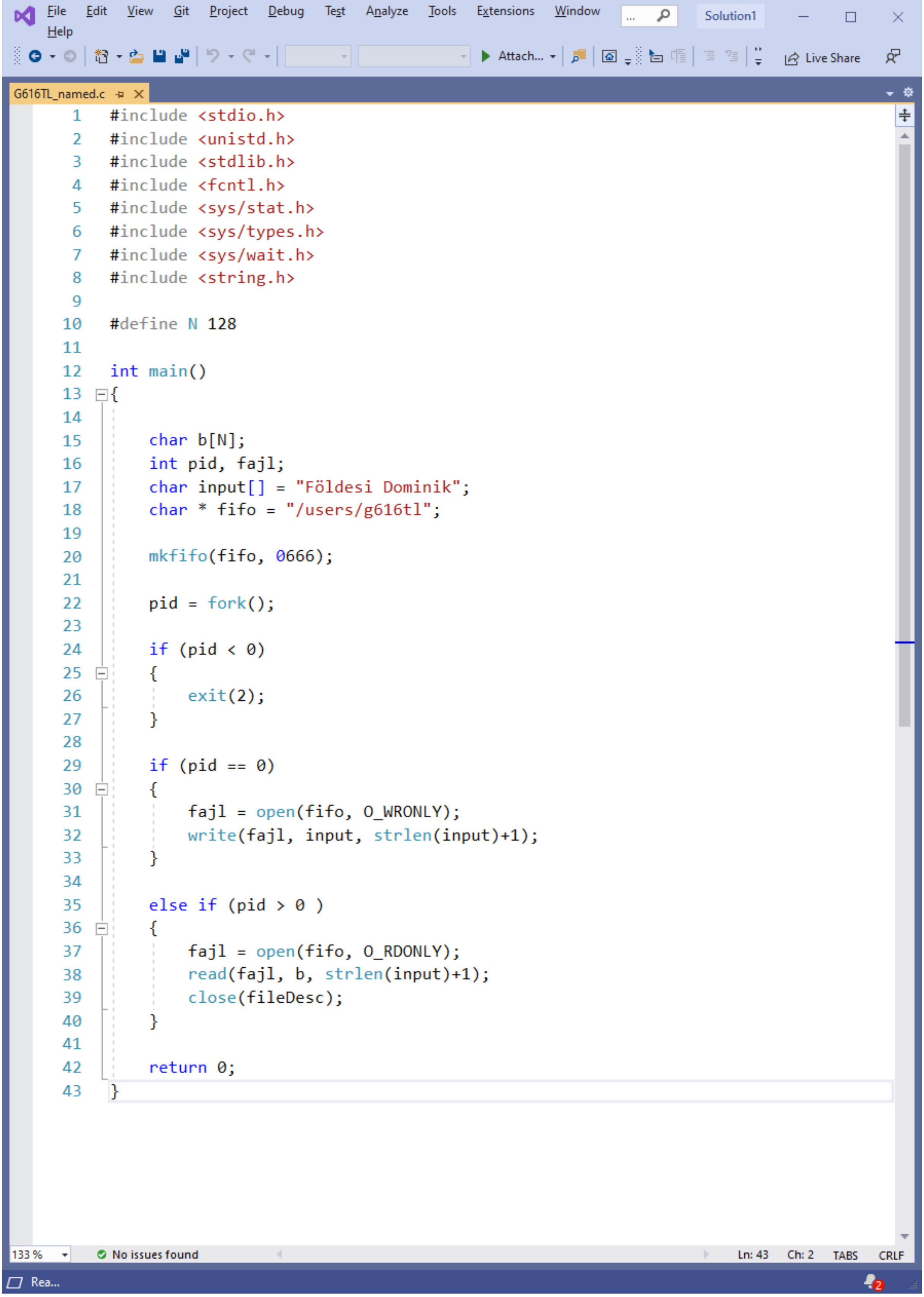
1.



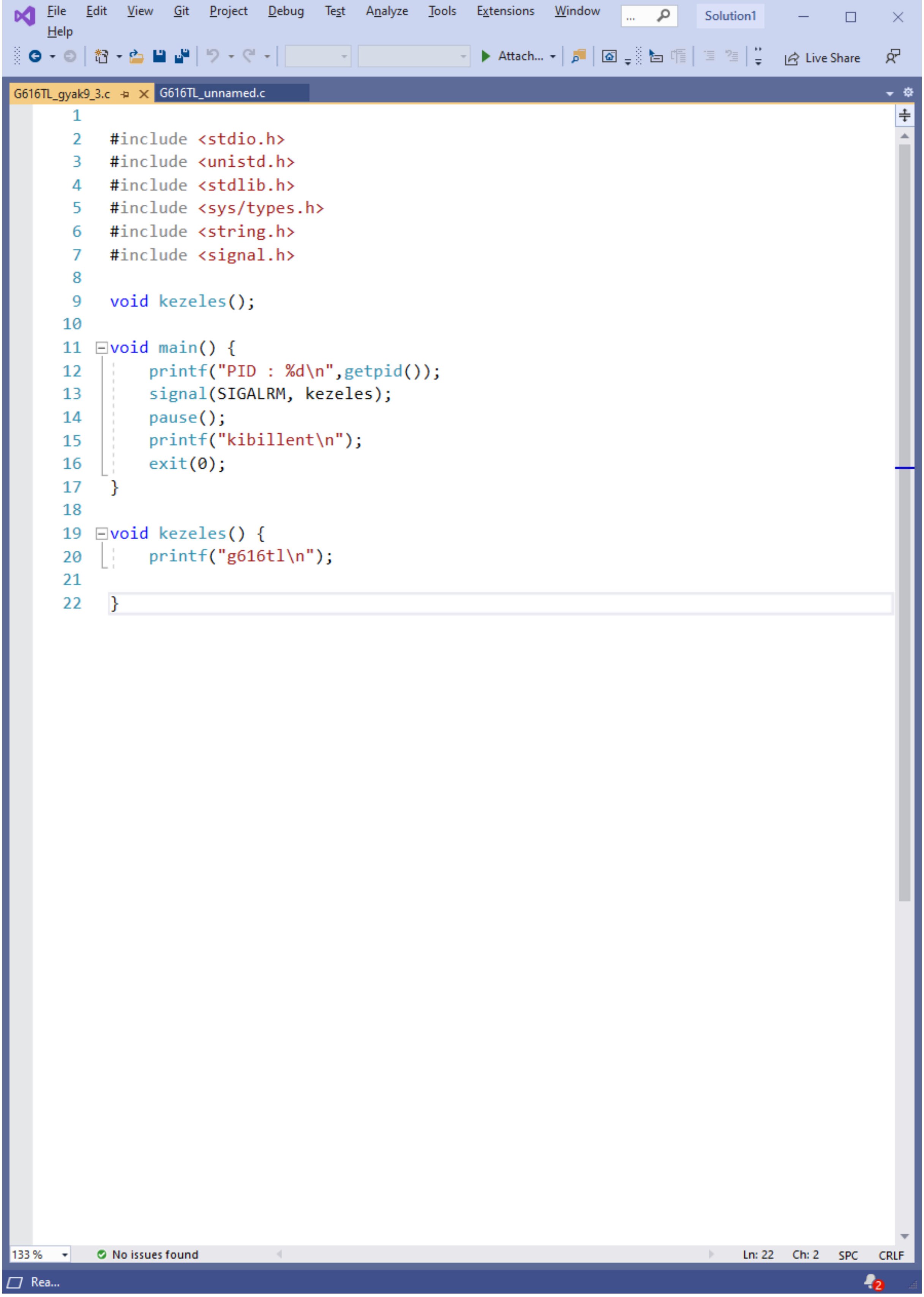
```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>

int main() {
    int n;
    pid_t childpid;
    char string[] = "XY G616TL\n";
    char read[80];
    int fd[1];
    pipe(fd);

    if((childpid = fork()) == -1) {
        perror("fork");
        exit(1);
    }

    if(childpid == 0) {
        close(fd[0]);
        write(fd[1], string, (strlen(string)+1));
        exit(0);
    } else {
        close(fd[1]);
        n = read(fd[0], read, sizeof(read));
        printf("kapott string: %s", read);
    }

    return(0);
}
```

2.



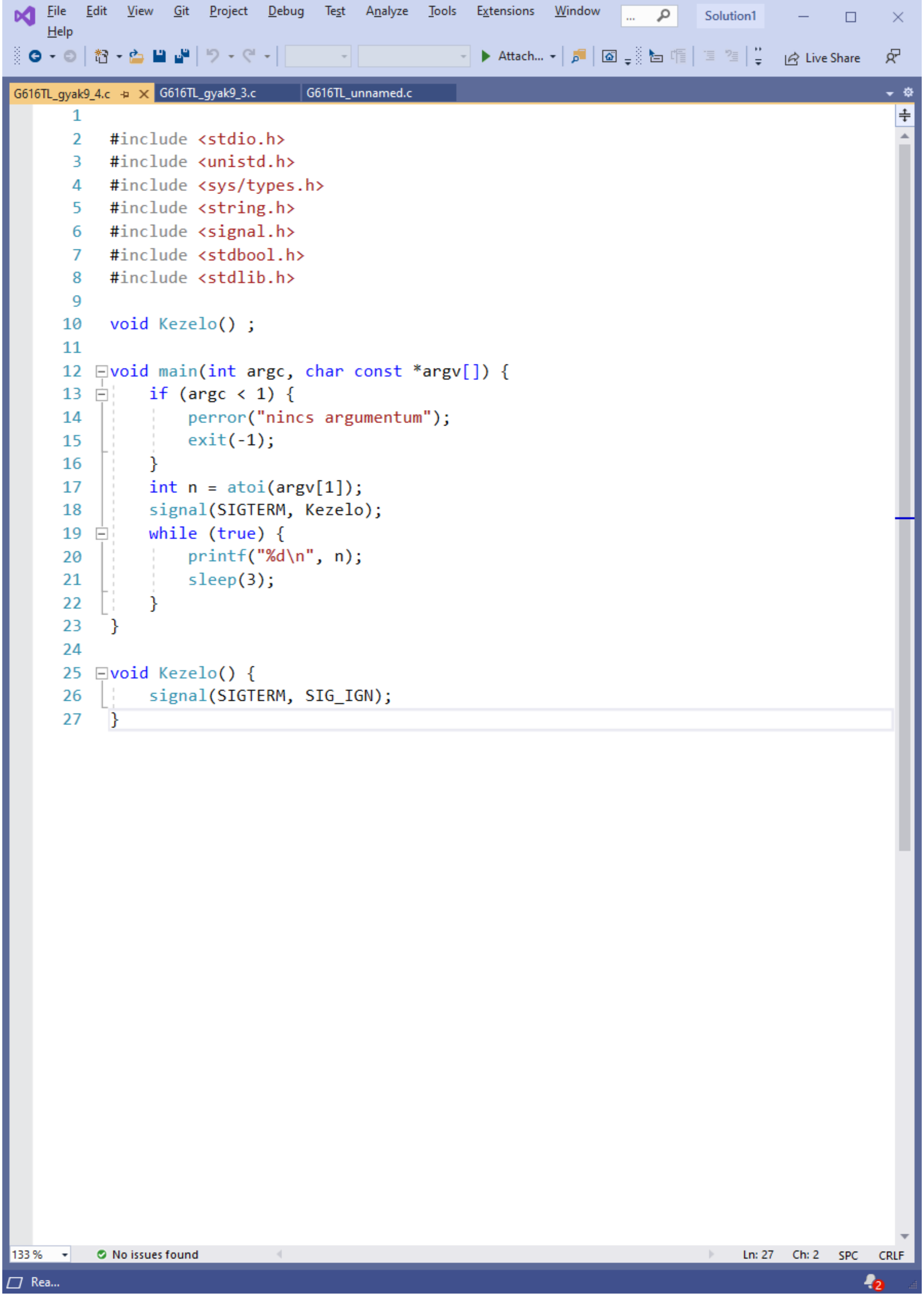
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>

#define N 128

int main()
{

    char b[N];
    int pid, fajl;
    char input[] = "Földesi Dominik";
    char * fifo = "/users/g616tl";

    mkfifo(fifo, 0666);

    pid = fork();

    if (pid < 0)
    {
        exit(2);
    }

    if (pid == 0)
    {
        fajl = open(fifo, O_WRONLY);
        write(fajl, input, strlen(input)+1);
    }

    else if (pid > 0 )
    {
        fajl = open(fifo, O_RDONLY);
        read(fajl, b, strlen(input)+1);
        close(fileDesc);
    }

    return 0;
}
```

3



```c

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <string.h>
#include <signal.h>

void kezeles();

void main() {
    printf("PID : %d\n",getpid());
    signal(SIGALRM, kezeles);
    pause();
    printf("kibillent\n");
    exit(0);
}

void kezeles() {
    printf("g616tl\n");

}
```

4.



```

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>
#include <signal.h>
#include <stdbool.h>
#include <stdlib.h>

void Kezelo() ;

void main(int argc, char const *argv[]) {
    if (argc < 1) {
        perror("nincs argumentum");
        exit(-1);
    }
    int n = atoi(argv[1]);
    signal(SIGTERM, Kezelo);
    while (true) {
        printf("%d\n", n);
        sleep(3);
    }
}

void Kezelo() {
    signal(SIGTERM, SIG_IGN);
}
```