1.)

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

int main()
{
        if(!system("dir"))
        {
                printf("\ndir hivas mukodik\n");
        }

        system("nemletezoparancs");

        return 0;
}
```

```
simon28@jerry:~/oprend/second_try/gyak5$ nano XUE9MHgyak1.c
simon28@jerry:~/oprend/second_try/gyak5$ gcc XUE9MHgyak1.c -o gyak1
simon28@jerry:~/oprend/second_try/gyak5$ ./gyak1
gyak1  XUE9MHgyak1.c

dir hivas mukodik
sh: 1: nemletezoparancs: not found
```

A program system() parancs segitségével kiadja a dir (létező) parancsot, amely kilistázza a futtatott programmal megegyező mappában lévő fájlok listáját. A „nemletezoparancs“ hívásakor jelzi hogy nem talalja a parancsot. Az előtte lévő „sh: 1:“ foggalmam sincs mit szimbolizál.

2.)

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

int main()
{
        char cmd[50];
        do {
                scanf("%s",cmd);
                system(cmd);
        }while(1)

        return 0;
}
```

```
simon28@jerry:~/oprend/second_try/gyak5$ gcc XUE9MHgyak2.c -o gyak2
simon28@jerry:~/oprend/second_try/gyak5$ ./gyak2
date
Sat Apr 24 09:59:14 CEST 2021
pwd
/home/stud2019/simon28/oprend/second_try/gyak5
who
more     pts/4        2021-04-21 15:44 (2a01:36d:1400:20a9:28fd:d3e0:70f0:a02)
vincze6  pts/5        2021-04-08 13:47 (2001:738:6001:500::4)
simon28  pts/6        2021-04-24 09:48 (178.143.13.226)
^\Quit
```

Beolvasunk a C nyelvhez tartozó scanf() függvényhívással egy „string“-et egy char tömbbe, majd végrehajtatjuk ezt a system() -el . A program gyorsbillentyűs megszakításig fut, bár egy if-el simán bele lehetett volna tenni hogy egy megadott beolvasott szövegre is leálljon.

3.) Parent:

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

int main()
{
        printf("Gyerek processz futtatasa: \n");
        system("./child");
        printf("Gyerek processz lefutott!\n");

        return 0;
}
```

Child:

```c
#include<stdio.h>

int main()
{
        for(int i=0; i < 5; i++)
        {
                printf("Simon Daniel, XUE9MH\n");
        }
        return 0;
}
```

Futtatás:

```
simon28@jerry:~/oprend/second_try/gyak5$ gcc parent.c -o parent
simon28@jerry:~/oprend/second_try/gyak5$ ./parent
Gyerek processz futtatasa:
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Gyerek processz lefutott!
```

Tudom lehetett volna (vagy valószinűleg úgy is kellett volna) fork()-val is, de így is megfelel a feladatkiírásnak és miért bonyolítsam a programot ha nem indokolt. Előző feladatokban a system() parancsot próbálgattuk így szerintem nem túl rossz megoldás ez sem.
Gyerek program for ciklussal kiirja a nevem és neptunkódom, szülő jelzi mikor futtassa a gyerek programot és azt is mikor lefutott, kettő közt a system utasítással futtassa az általam előre le-compile-olt gyerek programot.

4.)

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

int main()
{
        pid_t pid;
        if((pid = fork()) < 0)
        {
                perror("Fork error");
        } else
                if(pid == 0)
                        execl("./child","child",(char*)NULL);
        waitpid(pid,NULL,0);
}
```

```
simon28@jerry:~/oprend/second_try/gyak5$ nano XUE9MHgyak4.c
simon28@jerry:~/oprend/second_try/gyak5$ gcc XUE9MHgyak4.c -o gyak4
simon28@jerry:~/oprend/second_try/gyak5$ ./gyak4
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
Simon Daniel, XUE9MH
simon28@jerry:~/oprend/second_try/gyak5$
```

fork() hívás után lekezeltem az esetleges hibákat majd az execl(...) hívással futtattam az előző feladatban megírt child programot. A program megvárta míg lefutott a gyerek program (bár tippre waitpid nélkül is megvárta volna).

5.)

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

int main()
{
        int status = 7;
        pid_t pid;
        if((pid = fork()) < 0)
                perror("Fork error");
        else
                if(pid  == 0)
                        exit(status);

        waitpid(pid,&status,0);

        if(wait(&status) != pid)
                perror("wait error");
        if(WIFEXITED(status))
                printf("Normalis befejozedes, visszaadott ertek: %d\n",WEXITSTATUS(status));

        if((pid = fork()) < 0)
                perror("fork error");
        else
                if(pid == 0)
                        abort();

        waitpid(pid,&status,0);

        if(wait(&status) != pid)
                perror("wait error");
        if(WIFSIGNALED(status))
                printf("Abnormalis befejezodes, a szignal sorszama: %d\n",WTERMSIG(status));

}
```

```
simon28@jerry:~/oprend/second_try/gyak5$ gcc XUE9MHgyak5.c -o gyak5
XUE9MHgyak5.c: In function ‘main’:
XUE9MHgyak5.c:10:12: warning: implicit declaration of function ‘fork’ [-Wimplici
t-function-declaration]
  if((pid = fork()) < 0)
            ^~~~
simon28@jerry:~/oprend/second_try/gyak5$ ls
child    gyak1  gyak4  parent    XUE9MHgyak1.c  XUE9MHgyak4.c
child.c  gyak2  gyak5  parent.c  XUE9MHgyak2.c  XUE9MHgyak5.c
simon28@jerry:~/oprend/second_try/gyak5$ ./gyak5
wait error: No child processes
Normalis befejozedes, visszaadott ertek: 7
wait error: No child processes
Abnormalis befejezodes, a szignal sorszama: 6
simon28@jerry:~/oprend/second_try/gyak5$
```

Ezt egyszerűen nem tudom működésre bírni, több időt fordítottam erre a feladatra mint az előző négyre együtt, de még így sem értem a feladatleírásból, hogy hogy kellene. Programoztam valamit de ez nem ér semmit, nem tudom hogy a status-t hol kellene deklarálni és mivel kellene feltölteni. Kerestem az interneten is de nem találtam semmi hasznosat.