1. Adott egy rendszer (foglalási stratégiák), melyben a következő

- Szabad területek: 30k, 35k, 15k, 25k, 75k, 45k és
- Foglalási igények: 39k, 40k, 33k, 20k, 21k állnak rendelkezésre.

Határozza meg változó partíció esetén a következő algoritmusok felhasználásával: first fit, next fit, best fit, worst fit a foglalási igényeknek megfelelő helyfoglalást!

First fit	**30**	**35**	**15**	**25**	**75**	**45**
39	30	35	15	25	39, 36	45
40	30	35	15	25	39, 36	40, 5
33	30	33,2	15	26	39, 36	40, 5
20	20, 10	33,2	15	25	39, 36	40, 5
21	20, 10	33,2	15	21, 4	39, 36	40, 5

Next fit	**30**	**35**	**15**	**25**	**75**	**45**
39	30	35	15	25	39, 36	45
40	30	35	15	25	39, 36	40, 5
33	30	33, 2	15	25	39, 36	40, 5
20	30	33, 2	15	20, 5	39, 36	40, 5
21	30	33, 2	15	20, 5	39, 21, 18	40, 5

Best fit	**30**	**35**	**15**	**25**	**75**	**45**
39	30	35	15	25	75	39, 6
40	30	35	15	25	40, 35	39, 6
33	30	33, 2	15	25	40, 35	39, 6
20	30	33, 2	15	20, 5	40, 35	39, 6
21	21, 9	33, 2	15	20, 5	40, 35	39, 6

Worst fit	**30**	**35**	**15**	**25**	**75**	**45**
39	30	35	15	25	39, 36	45
40	30	35	15	25	39, 36	40 ,5
33	30	35	15	25	39, 33, 3	40, 5
20	30	20, 15	15	25	39, 33, 3	40, 5
21	21, 9	20, 15	15	25	39, 33, 3	40, 5

2. Gyakorló feladat: A feladat megoldásához először tanulmányozza Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (6.4)., azaz

Írjon C nyelvű programokat, ahol

- kreál/azonosít szemafor készletet, benne N szemafor-t. A kezdő értéket 0-ra állítja – semset.c,
- kérdezze le és írja ki a pillanatnyi szemafor értéket – semval.c
- szüntesse meg a példácskák szemafor készletét – semkill.c
- sembuf.sem_op=1 értékkel inkrementálja a szemafort – semup.c

semset.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */

    int semid,    /* semaphor set azonosito */
        nsems,    /* semaphorok szama a keszletben */
        semnum,   /* semapho szam a set-en belul */
        rtn;      /* visszateresi ertek */

      int semflg; /* flag */
    int semval;

    union semun {
        int val;
        struct semid_ds *buf;
        unsigned short  *array;
    } arg;
    int cmd;

int main()
{
    nsems = 1;
    semnum = 1;
    semflg = 00666 | IPC_CREAT;
    semval = 0;

    if (semid < 0 ) {perror(" semget hiba"); exit(0);}
      else printf("\n semid: %d ",semid);

    //int rtn = semctl(semid, semval, SETVAL, arg);

    semnum = 0;   /* 0-i semaphort azonositom */

    arg.val = 0;
    cmd = SETVAL; /* allitsd be a semaphor erteket */
      rtn = semctl(semid, semnum, cmd, arg); /* a semid-vel azonositott
                                  set 0-ik semaphorat ! */
```

```
    printf("\n set  rtn: %d ,semval: %d ",rtn,arg.val);
      printf("\n");

    return 0;
}
```

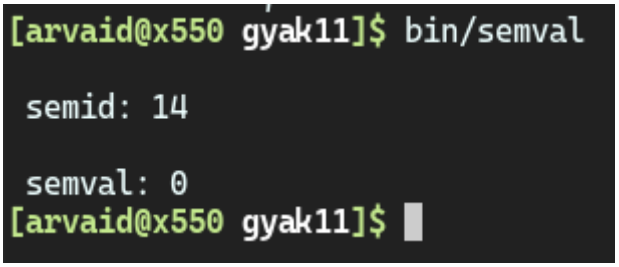
Futtatás eredménye:

```
[arvaid@x550 gyak11]$ bin/semset

 semid: 0
 set  rtn: -1 ,semval: 0
[arvaid@x550 gyak11]$
```

semval.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek */


      int semid,nsems,rtn;
      int semflg;
      int cmd;

    union semun {
        int val;
        struct semid_ds *buf;
        unsigned short  *array;
    } arg;

int main()
{
      nsems = 1;
      semflg = 00666 | IPC_CREAT;
      semid = semget (SEMKEY, nsems, semflg);
      if (semid < 0 ) {perror(" semget hiba"); exit(0);}
      else printf("\n semid: %d ",semid);
      printf ("\n");

      cmd = GETVAL;     /* E parancsra a semctl visszaadja a currens
                    semaphor erteket. Itt az rtn-be.  */
      rtn = semctl(semid,0, cmd, NULL);

      printf("\n semval: %d ",rtn);
      printf("\n");

}
```

Futtatás eredménye:



semup.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */

      int semid,nsems,rtn;
      unsigned nsops;   /* semop-ban ezzel adjuk meg, hany semaphore
                     strukturaval foglakozzon */
      int semflg;
      struct sembuf sembuf, *sop;


int main()
{
      nsems = 1;  /* Egy semaphore legyen */
      semflg = 00666 | IPC_CREAT;
      semid = semget (SEMKEY, nsems, semflg);
      if (semid < 0 ) {perror(" semget hiba"); exit(0);}
      else printf("\n semid: %d ",semid);
      printf ("\n");

      nsops = 1;        /* Egy operacio van */
      sembuf.sem_num = 0;     /* A 0-ik semaphor-ral foglakozunk */
      sembuf.sem_op = 1;      /* Inkrementaciot kerunk! */
      sembuf.sem_flg = 0666;  /* Flag beallitas */
      sop = &sembuf;          /* Igy keri a semop az argumentumot */
      rtn = semop(semid, sop, nsops);
            /* 0-val visszatero semop sikeres. */
      printf("\n up rtn: %d ",rtn);
      printf("\n");

}
```

Futtatás eredménye:

semkill.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */


      int semid,nsems,rtn;
      int semflg;
      int cmd;

    union semun {
        int val;
        struct semid_ds *buf;
        unsigned short  *array;
    } arg;

int main()
{
      nsems = 1;
      semflg = 00666 | IPC_CREAT;
      semid = semget (SEMKEY, nsems, semflg);
      if (semid < 0 ) {perror(" semget hiba"); exit(0);}
      else printf("\n semid: %d ",semid);
      printf ("\n");

      cmd = IPC_RMID;         /* Ez a megszuntetes parancsa */
      rtn = semctl(semid,0, cmd, arg);

      printf("\n kill rtn: %d ",rtn);
      printf("\n");

}
```

Futtatás eredménye: