1.

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

int main(int argc, char **argv)
{

  int n, i, id;
   char name[20];
   if (argc !=2)
   {
       print("argument number error\n");
       exit (1);
   }
  n = atoi(argv[1]);
   typedef struct{
     int id;
     char lname[20];
     char fname[20];
   } data;

   data *list;
   list = (data*) calloc( n,sizeof(data) );
   for (i=0; i < n; i++)
   {
      printf("Enter  first name: \n");
      scanf("%s",name);
      strcpy (list[i].fname, name);
      printf("Enter  last name: \n");
      scanf("%s",name);
      strcpy (list[i].lname, name);
      printf ("Enter ID number: \n");
      scanf("%d", &id);
      list[i].id =id;
   }

    printf ("YOUR STUDENT LIST\n");
    for (i=0; i < n; i++)
    {
         printf("%s   %s   %d \n",list[i].fname, list[i].lname, list[i].id);
    }
   free( list );
   return 0;

}
```

2.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
        pid_t pid, pid1, ppid;
        int i, endID, status;
        pid = fork(); /*create the first child */
        if (pid == 0) /* code for first child */
        {
                pid1 = fork(); //create a grandchild by the first child
                if (pid1 >0)
                {
                        for (i=0; i<100; i++)
                        {
                                printf("I am your child with ID = %d \n", getpid());
                                sleep(1);
                        }
                        _exit(0);
                }
                else
                {
                        ppid = getppid();
                        while (1)
                        {
                                if (getppid()==ppid)
                                {
                                        printf("I am your grandchild with ID = %d \n", getpid());

                                        sleep(1);
                                }
                                else
                                        _exit(0);
                        }
                }
        }
        else
        {
                while (1)
                {
                        endID=waitpid(pid, &status, WNOHANG|WUNTRACED);
                        if (endID==0) //child still running
                        {
                                printf("I am your parent with ID= %d\n",getpid());
                                sleep(1);
                        }
                        else
                        {
                                printf("Now my job is over \n");
                                exit(0);
                        }
                }
        }
        return 0;
}
```

3.

```
#include <fcntl.h>
#include  <stdio.h>
#include <sys/stat.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
       int input, output, nbyte, size;
       int i;
       char buff[1];

       input = open(argv[1], O_RDONLY);
       size = lseek(input, 0, SEEK_END); // check size of input
       pid_t  pid;

       pid = fork(); /* create a child */
       umask(0);
       if (pid == 0) /* child process */
       {
              if ((output = open("child.txt", O_WRONLY|O_CREAT, S_IREAD|S_IWRITE)) == -1)
              {
                     printf("Output File Create Error");
                     exit (1);
              }
       }
       else
       {
              if ((output = open("parent.txt", O_WRONLY|O_CREAT, S_IREAD|S_IWRITE)) == -1)
              {
                     printf("Output File Create Error");
                     exit (1);
              }
       }
       // now either child or parent run same code on its own space!!!!
       for (i =0; i< size; i++)
       {

              if (pread(input, buff, 1, i)!= 1)
              {
                     printf("Read Error");
                     exit (2);
              }
              if (write(output, buff, 1) != 1)
              {
                           printf("Write Error");
                           exit (3);
              }

       }
       exit(0);
}
```