17. Write a C program that behaves like a shell (command interpreter). It has its own prompt say "NewShell$". Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.

i) list f <dirname> - print name of all files in directory

ii) list n <dirname> - print number of all entries

iii) list i<dirname> - print name and inode of all files

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/stat.h>
#include<dirent.h>
#include<fcntl.h>

void list(char op,char *dn)
{
     DIR *dirp;
     struct dirent *e;
     int dc=0,fc=0;
     dirp=opendir(dn);
     if(dirp==NULL)
     {
          printf("directory %s not found \n",dn);
          return;
     }
     switch(op)
     {
          case 'f' :
               while(e==readdir(dirp))
               {
                    if(e->d_type==DT_REG)
                         printf("%s\n",e->d_name);
               }
          break;
          case 'n' :
               while(e==readdir(dirp))
```

```
                {
                     if(e->d_type==DT_DIR)
                          dc++;
                     if(e->d_type==DT_REG)
                          fc++;
                }
                printf("%d file(s)  %d dir(s)\n",fc,dc);
          break;
          case 'i' :
                while(e==readdir(dirp))
                {
                     if(e->d_type==DT_REG)
                          printf("%s\t%ld\n",e->d_name,e-
>d_fileno);
                }
          break;
          default :
                printf("Invalid Option");
          break;
     }
}

int main()
{
     char buff[80],t1[30],t2[30],t3[30],t4[30];
     int pid,n;
     while(1)
     {
          printf("\nmyshell $");
          fflush(stdin);
          fgets(buff,80,stdin);
          n=sscanf(buff,"%s%s%s%s",t1,t2,t3,t4);
          switch(n)
          {
                case 1 :
                     pid=fork();
                     if(pid==0)
                          execlp(t1,t1,NULL);
                     else
                          wait();
                break;
                case 2 :
                     pid=fork();
                     if(pid==0)
```

```
                    execlp(t1,t1,t2,NULL);
               else
                    wait();
          break;
          case 3 :
               if(strcmp(t1,"list")==0)
                    list(t2[0],t3);
               else
               {
                    pid=fork();
                    if(pid==0)
                         execlp(t1,t1,t2,t3,NULL);
                    else
                         wait();
               }
          break;
          case 4 :
               pid=fork();
               if(pid==0)
                    execlp(t1,t1,t2,t3,t4,NULL);
               else
                    wait();
          break;
     }
  }
}
```