```
/* Author(s): Please put your student name(s) & section here.
 *
 * This is a lab6.c the csc60mshell
 * This program serves as a skeleton for starting for lab 6.
 * Student is required to use this program to build a mini shell
 * using the specification as documented in direction.
 * Date: Fall 2016
 */

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

#define MAXLINE 80
#define MAXARGS 20
#define MAX_PATH_LENGTH 50
#define TRUE 1

/* function prototypes */
/* void process_input(int argc, char **argv); */
int parseline(char *cmdline, char **argv);
/* void handle_redir(int count, char *argv[]); */

/* ---------------------------------------------------------------- */
/*                The main program starts here                      */
/* ---------------------------------------------------------------- */
int main(void)
{
   char cmdline[MAXLINE];
   char *argv[MAXARGS];
   int argc;
   int status;
   pid_t pid;

   /* Loop forever to wait and process commands */
   while (TRUE) {
           /* Print your shell name: csc60mshell (m for mini shell) */
           printf("FillInThisSpace> ");

           /* Read the command line */
           fgets(cmdline, MAXLINE, stdin);

           /* Call parseline to build argc/argv: their limits declared above */
```

```
            /* If user hits enter key without a command, continue to loop again at the beginning */
            /*  Hint: if argc is zero, no command declared */
            /*  Hint: look up for the keyword "continue" in C */

            /* Handle build-in command: exit, pwd, or cd  */

//......................IGNORE.......................
//          /* Else, fork off a process */
//          pid = fork();
//          switch(pid)
//          {
//             case -1:
//                     perror("Shell Program fork error");
//                exit(1);
//             case 0:
//                     /* I am child process. I will execute the command, call: execvp */
//                     process_input(argc, argv);
//                     break;
//             default:
//                     /* I am parent process */
//                     if (wait(&status) == -1)
//                        perror("Shell Program error");
//                     else
//                        printf("Child returned status: %d\n",status);
//                     break;
//          } /* end of the switch */
//..............end of the IGNORE above........................

   } /* end of the while */
} /* end of main */
/* ----------------------------------------------------------------- */
/*                parseline                                  */
/* ----------------------------------------------------------------- */
/* parse input line into argc/argv format */

int parseline(char *cmdline, char **argv)
{
   int count = 0;
   char *separator = " \n\t";

   argv[count] = strtok(cmdline, separator);
   while ((argv[count] != NULL) && (count+1 < MAXARGS)) {
            argv[++count] = strtok((char *) 0, separator);
   }
   return count;
}
/* ----------------------------------------------------------------- */
/*                process_input                                */
/* ---------------------------------------------------------------- */
void process_input(int argc, char **argv)
{
   /* Step 1: Call handle_redir to deal with operators:    */
   /* < , or  >, or both                                   */

   /* Step 2: perform system call execvp to execute command      */
   /* Hint: Please be sure to review execvp.c sample program      */
   /* if (........ == -1) {                    */
   /*  perror("Shell Program");                */
   /*  _exit(-1);                              */
   /* }                                        */

}
/* ---------------------------------------------------------------- */
//void handle_redir(int count, char *argv[])

/* Put your code here.  See pseudo-code in assignment directions     */

/* ---------------------------------------------------------------- */
```