```
// Remove the comments below that are high-lighted
/* Author(s): Please put your student name(s).
 *
 * This is lab9.c the csc60mshell
 * This program serves as a skeleton for doing labs 9, 10, 11.
 * Student is required to use this program to build a mini shell
 * using the specification as documented in direction.
 * Date: Fall 2019
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>

/* Define Section */
#define MAXLINE 80
#define MAXARGS 20
#define MAX_PATH_LENGTH 50
#define TRUE 1

/* function prototypes */
int parseline(char *cmdline, char **argv);
//The two function prototypes below will be needed in lab10.
//Leave them here to be used later.
/* void process_input(int argc, char **argv); */
/* void handle_redir(int argc, 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: csc60msh (m for mini shell) */
      printf("FillInThisSpace> ");

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

      /* Call parseline to build argc/argv */
      /* 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  */
     /* Put the rest of your code here */

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

    /* strtok searches for the characters listed in separator */
    argv[argc] = strtok(cmdline, separator);

    while ((argv[argc] != NULL) && (argc+1 < MAXARGS))
      argv[++argc] = strtok((char *) 0, separator);

    return argc;
}
/* ----------------------------------------------------------------- */
/*                 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       */
    // The exec all goes here
    /* if (........ == -1) {                                         */
    /*    fprintf(stderr, "Error on the exec call\n");               */
    /*    _exit(EXIT_FAILURE);                                       */
    /* }                                                             */

// }
/* ---------------------------------------------------------------- */
//void handle_redir(int count, char *argv[])
/* ---------------------------------------------------------------- */
//code goes here
/* ---------------------------------------------------------------- */
```