Your shell program must provide services to support the following commands and features:

Note: The commands are shown in uppercase font, but don't have to be programmed that way. The parameters to the commands are enclosed between the symbols < and >, but the symbols are not actually part of the parameters.

a. **(5 marks)** `STOP`: Terminates execution of the current `myshell` session.

b. **(5 marks)** `SETSHELLNAME` <*shell_name*>: Sets the shell name in the `myshell` command prompt to <*shell_name*>. For example, a sample command prompt is shown below:

   ```
   myshell>
   ```

   In this command prompt, there are two parts. The first part, `myshell`, is the shell name. The second part, >, is the terminator. If no shell name is defined, `myshell` should be the default shell name.

c. **(5 marks)** `SETTERMINATOR` <*terminator*>: Sets the terminator in the `myshell` command prompt to <*terminator*>. If no terminator is defined, `myshell` should use > as the default terminator.

d. **(10 marks)** `NEWNAME` <*new_name*> | <*new_name*> <*old_name*>: Manages the alias list. The first option deletes a previously defined alias. The second option defines an alias for another command. For example, the command `NEWNAME mymove` deletes the alias for `mymove`, and the command `NEWNAME mycopy cp` defines `mycopy` as the alias for the `cp` command. If an alias for a command already exists, then the new alias replaces the old alias. The maximum number of aliases in the alias list should be set to 10 as the default.

e. **(10 marks)** `LISTNEWNAMES`: Outputs all the aliases that have been defined. Each pair of names should be shown on one line. For example, the possible aliases for a few commands are shown below:

   ```
   mycd cd
   ```

   ```
   mycopy cp
   ```

f. **(10 marks)** `SAVENEWNAMES` <*file_name*>: Stores all currently defined aliases in the file <*file_name*>.

g. **(10 marks)** `READNEWNAMES` <*file_name*>: Reads all aliases in the file <*file_name*> and adds them to the aliases defined in the current session. If a duplicate is found in the file <*file_name*>, it should be ignored.

h. **(5 marks)** <*UNIX_command*>: Executes the UNIX command <*UNIX_command*>, corresponding to any valid UNIX command. One approach to implementing this is to use the `system` function. If the first token on a command line is not a built-in command, assume that it is a UNIX command.

i. **(5 marks)** HELP: shows all built-in commands supported by myshell.

j. **(10 marks)** Error handling: Your approach should effectively identify and recover from errors. For example, bad input and/or the inability to execute a command should not cause `myshell` to crash.

Note: You must handle all the built-in commands with exactly the same syntax as shown above. Thus, an important part of the `myshell` program will be to parse commands entered at the command prompt to break them down into their component parts. Once a command has been parsed, the component parts can be checked to ensure that a valid command has been entered and that it adheres to the required syntax.

For this problem, the results are worth 75 marks out of the 100 marks available. The breakdown of these marks is shown above. Demonstrate that your shell works and that it can handle all of the requirements described above, except for the last one (i.e., the error handling) as you will demonstrate that in Part 2. So, this part 1 will be worth 65 marks out of the 75 marks available. Your demonstration should be captured in a script file and follow the sequence of commands given below:

```
ls -l
SETSHELLNAME mysh
who
NEWNAME mycopy cp
```
press `Return` *with a blank command line*
```
NEWNAME mycat cat
SAVENEWNAMES myaliases
LISTNEWNAMES
mycat myaliases
mycopy /etc/motd myfile
mycat myfile
cat myfile
NEWNAME mydel rm
ls
mycopy myfile
ls
SAVENEWNAMES myaliases
mycat myaliases
SETSHELLNAME myshell
LISTNEWNAMES
READNEWNAMES myaliases
mycat
```

PART 3 Error Handling 10 points

Finally, demonstrate that your shell can handle errors (i.e., the last item in Part 1). This will be worth 10 marks out of the 75 marks available. In order to receive all 10 marks, you must be able to handle at least 5 different kinds of errors. Your demonstration should be captured in a script file.