Editing and Browsing code exercise solutions

4.1 Vim Exercises

1. **1st question solution**

```
ignitarium@ignitarium-Lenovo-ThinkBook-14-IML:~$ vim .vimrc
ignitarium@ignitarium-Lenovo-ThinkBook-14-IML:~$ cat .vimrc
set tabstop=4
set shiftwidth=4
set expandtab
set autoindent
ignitarium@ignitarium-Lenovo-ThinkBook-14-IML:~$
```

---

2. **2nd question solution:**

I have created a C file named ‘a.c’ which consists of 4 dummy functions, each function having nested if loops in it. Here variables and arguments are multi-user names.

```
ignitarium@IGN-BLR-LP-215:~/editing_browsing$ cat a.c
#include<stdio.h>

int func1(int arg1, int kat) {
int var1=100,var2=200,var3=300;
if(var1 != var2) {
if(var2 != var3) {
printf("%d",&var3);
}
}
}

int func2(int arg2, int arg3) {
int var1=100,var2=200,var3=300;
if(var1 != var2) {
if(var2 != var3) {
printf("%d",&var3);
}
}
}

int func3(int arg4) {
int var1=100,var2=200,var3=300;
if(var1 != var2) {
if(var2 != var3) {
printf("%d",&var1);
}
}
}

int func4(int qwer) {
int var1=100,var2=200,var3=300;
if(var1 != var2) {
if(var2 != var3) {
printf("%d",&var2);
}
else {
printf("%d",&var1)
}
}
}

ignitarium@IGN-BLR-LP-215:~/editing_browsing$
```

---

3. **3rd question solution:**

➔ Go to the end of the file: Press Esc and then Shift + G.
➔ Go to the start of the file: Press G key twice.
➔ Go to starting brace of first function:
➔ Go to the end brace:
➔ Go to the start of the line of function prototype:
➔ Go to starting parenthesis:
➔ Go to first argument:
➔ Go to last letter of first argument:

---

4. **4th question solution:**

➔ To put marker at start of each function, go to the location where you want to put marker and enter mx , where x represents single character id for new marker, similarly add new markers at start of each function.
➔ To move from one marker to other, press ‘ along with character id of that marker

---

5. **5th question solution:**

➔ To horizontally split try, “CTRL-spacebar h s” , then the window will split into two horizontally.
➔ In another window open a new file “b.c” and copy 1st and 3rd function from a.c into b.c. copy each block separately then save and close b.c.

---

6. **6th question solution:**

➔ Press v to switch to VISUAL LINE mode and highlight all lines of function 1 and 2 by pressing j. Then press < to left shift the selected lines
➔ Then again repeat the same and press = to correct the indentation

---

7. **7th question solution:**

a) :s/^*\(\^*)
b) :6,20s/int/unsigned int/

8. **8th question solution:**

➔ Go to insert mode and add a new variable in function 1.
➔ Then type esc q followed by a.
➔ q indicates recording and a indicates storing recordings in register a.
➔ Then type escape yy to copy the newly added variable.
➔ Now paste the variable using p then type q to quit recording
➔ Now type 3@a to repeat the same for the other three functions.

---

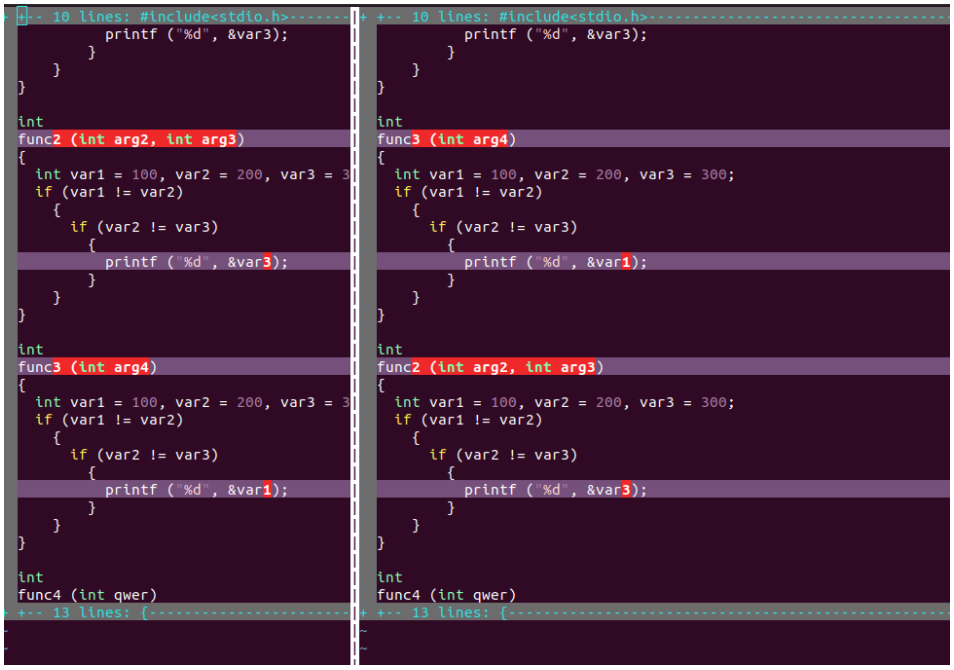
9. **9th question solution:**

➔ To vertically split try, “CTRL-spacebar v s” , then the window will split into two vertically.
➔ In another window open a new file “b.c” and copy the 2nd and 4th function from a.c, then save and close b.c. close a.c

---

10. **10th question solution:**

➔ Use vimdiff to find the difference between a.c and b.c



---

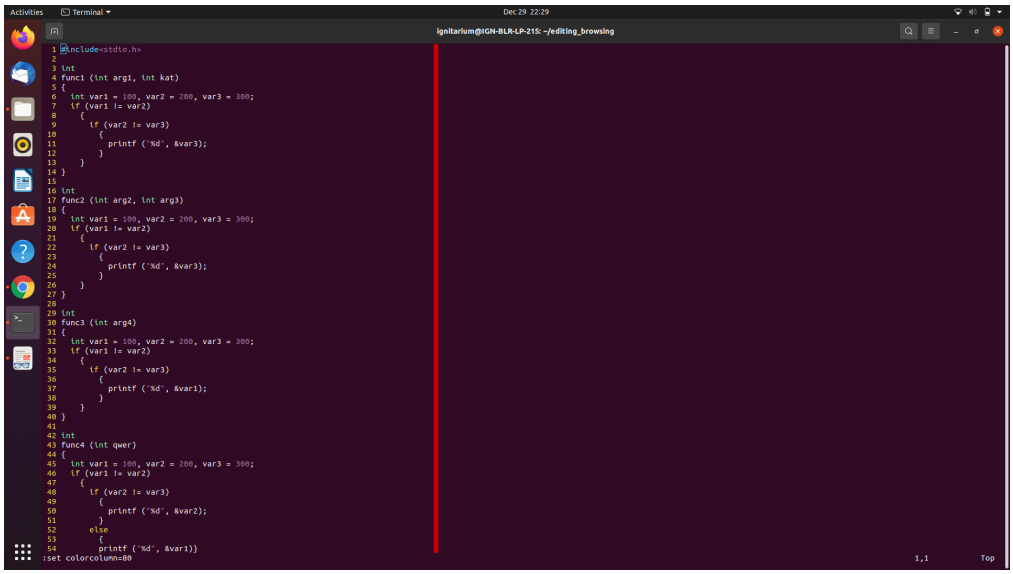
11. 11th question solution:

A. Show number lines:

➔ Press Esc key to switch to command mode, then press : and enter set number, this will set line numbers to the given file.

B. Show a vertical line at 80 column:

➔ Press Esc key to switch to command mode, then press : and enter set colorcolumn=80 , this will set line at column 80.



C. Execute a shell command and come back:

➔ Press Esc to go to command mode and press : and enter any shell command.

➔ Press Enter key to execute and come back to editor mode.

D. Symbol to refer to the file being edited in the command line: ls

E. To compile current file in vim command line: type :! gcc filename.c then type terminal and execute using ./a.out

F. To enable spell check: go to command mode and type “**:set spell spelllang=en_us**”

4.2 GNU Indent exercises

1. **1st question solution**

   a) Use -in command to set indentation to n

   b) set tabstop=2 shiftwidth=2 expandtab

   should do the trick. If you already have tabs, then follow it up with a nice global RE to replace them with double spaces.

   If you already have tabs you want to replace,

   :retab

---

2. **2nd question solution**

C code before running gnu-indent

```c
#include<stdio.h>

int func1(int arg1, int kat) {
    int var1=100,var2=200,var3=300;
        if(var1 != var2) {
    if(var2 != var3) {
        printf("%d",&var3);
}
}
}

int func2(int arg2, int arg3) {
    int var1=100,var2=200,var3=300;
            if(var1 != var2) {
    if(var2 != var3) {
                    printf("%d",&var3);
}
}
}

int func3(int arg4) {
    int var1=100,var2=200,var3=300;
    if(var1 != var2) {
            if(var2 != var3) {
    printf("%d",&var1);
}
}
}
                        int func4(int qwer) {
    int var1=100,var2=200,var3=300;
            if(var1 != var2) {
    if(var2 != var3) {
    printf("%d",&var2);
}
else {
    printf("%d",&var1);
}
}
}
```

after running gnu-indent

```c
func1 (int arg1, int kat)
{
  int var1 = 100, var2 = 200, var3 = 300;
  if (var1 != var2)
    {
      if (var2 != var3)
        {
          printf ("%d", &var3);
        }
    }
}

int
func2 (int arg2, int arg3)
{
  int var1 = 100, var2 = 200, var3 = 300;
  if (var1 != var2)
    {
      if (var2 != var3)
        {
          printf ("%d", &var3);
        }
    }
}

int
func3 (int arg4)
{
  int var1 = 100, var2 = 200, var3 = 300;
  if (var1 != var2)
    {
      if (var2 != var3)
        {
          printf ("%d", &var1);
        }
    }
}

int
func4 (int qwer)
{
  int var1 = 100, var2 = 200, var3 = 300;
  if (var1 != var2)
    {
      if (var2 != var3)
        {
          printf ("%d", &var2);
        }
      else
        {
          printf ("%d", &var1);
        }
    }
}
ignitarium@ignitarium-Lenovo-ThinkBook-14-IML:~/Downloads/tra
```

THANK YOU