1. (5 pt.)

   1)
   d: directory , -: regular file

   2)
   rwx for user, rx for group x for other

   3)
   chmod o+x snap.doc

   4)
   chmod g–r csc350

   5)
   ```
      % ls -l | wc
   Output of command ls -l are input for command wc (word count)
      % ls -l > wc
   Save output of command ls -l to file named wc
   ```

2. (3 pt.)
   - System call – totally controlled by OS, unbuffered I/O
   - Library function – controlled by program itself, buffered I/O.

3. (2 pt.)
   If a sticky bit for an executable file is set, then the first time the program was executed, a copy of the executable file was saved into swap area when the process terminate. If this process becomes active again, it will be loaded from the swap area in the memory.

4. (5 pt)

```
foo w -r  -rw
bar -r - w -rw
```

5. (15 pt.)

```
int myatoi(char *str)
{
    int num =0;
    int i =0;
    int negative = 0;
    if (str[0]== '-') //for negative number
    {
           i =1;//number start from the second charcter
           negative =1;
    }
    if (str[0]== '+') // possitive number with + sign
           i =1; //number start from the second charcter

    while (str[i]!='\0')
    {
           if ((str[i]>='0')&&(str[i]<='9'))
           {
                num = 10 * num + (str[i] - '0');
                i++;
           }
           else
           {
                perror("USAGE: input error");
                exit (1);
           }

   }
   if (negative == 1)
                      num = num * -1;
   return num;
}
```

6. (10 pt.)

```
char* myitoa(int x)
{
    static char buffer[22]; // Enough to hold integer in base 10
    int i = 20; // Start from the end of buffer
    int neg = 0;

    if (x < 0) {
        neg = 1;
        x = -x;
    }

    while (x > 0)
    {
           buffer[i] = x % 10 + '0'; // Convert to ASCII
           i--;
           x= x/10;
    }

    if (neg) {
        buffer[i] = '-';
        i--;
    }
    return &buffer[i+1];
}
```

7. (10 pt.)

```
#include <stdio.h>
#include <stdlib.h>
int myatoi(char *);
int main(int argc, char *argv[])
{
  int i, sum;
  sum =0;
  if (argc <= 1)// argument must be at least two or more
    {
      printf("argument number error \n");
      exit(1);
    }

  for (i=1; i<argc; i++)
  {
     if (myatoi(argv[i])%2 == 0)
          continue;
    sum = sum + myatoi(argv[i]);
  }
  printf("The sum of odd argument is %d\n", sum);
  return 0;
}
```

8. (10 pt.)

```
#!/usr/bin/bash
n=1
sum=0

if [ $# -eq 0 ]; then
   echo " No Arguments"
   exit 1
fi

for n in $*;
do
   let t=n%2
   if [ $t -eq 0 ]; then
          continue
     fi
   let sum=sum+$n
done
echo "Sum of Arguments is $sum"

exit 0
```

9. (5 pt.).

- Text editor- create a source code (ASCII)
- Preprocessor – include header files and create Modified Source code
- Compiler – compile modified source code and create binary object code
- Linker – link libraries to object code and create a executable code.

10. (5 pt.) What will be displayed for each of the following sequences of shell commands?
    a. $x
    b. you

11. (5 pt.)

How are you is going to print to STD_OUT
After dup2, STD_OUT descriptor will close and it is redirecting to fd.
I am fine Tank you will write to file my.txt

12. (15 pt.)

.

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

int main(int argc, char **argv)
{
      int in, out, i;  //file descriptors of files
      char c; //currently read character
      off_t offset; //current offset
      if (argc != 2){ //argument error check
        perror("Argument number error");
        exit (1);
      }
      if ((in = open(argv[1], O_RDONLY)) == -1){//input file error check
            perror("Input file error");
            exit (1);
      }
      umask(0); //clear mask
      out = open("mirrors", O_WRONLY|O_CREAT, 00600); //rw-------
      //get a size of input file
      offset = 0;
      while (read(in, &c, 1) >0)
            offset++;
      //read reverse order from input file and write to outputfile
      while(offset > 0){
            pread(in, &c, 1, offset-1); // read in reverse order
            write(out, &c, 1);  //write to output file
            offset--; // offset to previous character
      }

      write(out, "||",2); //write middle of mirror
      offset =0; // to make sure back to the begin
      while (pread(in, &c, 1, offset) > 0){
            write(out, &c, 1);
            offset++;
      }

      //close open files
      close(in);
      close(out);
      exit(0);
}
```

13. (5 pt)
- **vari**
- **Beauty and Beast**
- **$vari**
- **$vari**
- **$Beauty and Beast**

14. (5 pt.)

-
  ```
  gcc –c Fred.c
  gcc –c Bill.c
  ```
- By using gcc compiler, create test.o.
  ```
  gcc –c test.c
  ```
-
  ```
  ar crv libBF.a bill.o fred.o
  ```
-
  ```
  gcc –o foo foo.c lBF
  ```