1.

```
/*a) convert string to number */
int st_to_int(char *str)
{
  int num =0;
  int i =0;
  while (str[i]!='\0')
  {
      num = 10 * num + (str[i] - '0');
      i++;
  }
  return num;
}
void reverse(char str[], int length)
{
    int start = 0;
    int end = length -1;
     char tmp;
    while (start < end)
    {
           tmp=str[start];
           str[start]=str[end];
           str[end]=tmp;
           start++;
           end--;
    }
}
/* b) convert integer to c-string */
void int_to_st(char *str, int num)
{
    int i = 0;
    int isNegative =0;
    /* Handle 0 /
    if (num == 0)
    {
        str[i++] = '0';
        str[i] = '\0';
    }
   // Process individual digits backward!
    while (num != 0)
    {
        int rem = num % 10;
        str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';
        num = num/10;
    }
      str[i] = '\0'; // end of c-string terminator
    // Reverse the string
    reverse(str, i);
}
```

2.

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

3.

```
#include <stdio.h>
#include <stdlib.h>
int st_to_int(char *);
void 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 (st_to_int(argv[i])%2))
          continue;
    sum = sum + st_to_int(argv[i]);
  }
  printf("The sum of argument is %d\n", sum);
  return;
}

/* convert numberical c-string to number */
int st_to_int(char *str)
{
  int num =0;
  int i =0;
  int negative = 0;
  if (str[0]== '-')
  {
         i =1;
          negative =1;
  }
  while (str[i]!='\0')
  {
      num = 10 * num + (str[i] - '0');
      i++;
  }
  if (negative == 1)
          num = num * -1;
  return num;
}
```

4.

```
#!/bin/sh
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 1 ]; then
           continue
      elif [ $t -eq -1 ]; then
           continue
      fi
   let sum=sum+$n
done
echo "Sum of Arguments is $sum"

exit 0
```

5.

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

6. (5 pt.) What will be displayed for each of the following sequences of shell commands?
   a. $W
   b. K

7.

foo: r-rw-rw-
bar: -w—w-rw-

8.

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
       int InFileDes, OutFileDes;  /* file descriptors of files*/
       char curChar; /* currently read character */
       if (argc != 3)
       {
               printf("Argument number error \n");
               exit (1);
       }
       InFileDes = open(argv[1], O_RDONLY); /* open input file read only */
       if (InFileDes == -1)
       {
               printf("Input file error \n");
               exit (1);
       }
       umask(0);
       /* open output file, if doesn't exist create it with permis. rw-rw-rw-*/
        OutFileDes = open(argv[2], O_WRONLY|O_CREAT,0666);
       if (OutFileDes == -1)
       {
              printf("Output file error \n");
              exit (1);
       }
       dup2(OutFileDes, 1);
       while(read(InFileDes, &curChar, 1)> 0)
       {
              if ((int)curChar == 10) /* if character is next line */
              {
                            printf("\n");
                            continue;
              }
              printf("%d ",curChar);
       }

       close(InFileDes);
       close(OutFileDes);
       exit(0);
}
```

9.

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

int main(int argc, char *argv[])
{
  int InFileDes, OutFileDes;  //file descriptors of files
  char curChar; //currently read character
  off_t offset; //current offset
  if (argc != 3)
  {
       printf("Argument number error \n");
       exit (1);
  }
  InFileDes = open(argv[1], O_RDONLY); //open input file
  if (InFileDes == -1)
  {
       printf("Input file error \n");
       exit (1);
  }
  umask(0); //clear mask
  //open output file, if doesn't exsit with rw_rw_rw_
  OutFileDes = open(argv[2], O_WRONLY|O_CREAT,0666);
  //set offset to end of input file
  offset = lseek(InFileDes, -1, SEEK_END)+1;
  while(offset > 0) //while offset is not beginning of input file
  {
      read(InFileDes, &curChar, 1); //read each char of input file
       if ((curChar<'0')||(curChar >'9'))
            write(OutFileDes, &curChar, 1);
       lseek(InFileDes, -2, SEEK_CUR); //move offset
       offset--; //decrement offset
  }
  //close open files
  close(InFileDes);
  close(OutFileDes);
  exit(0);
}
```

10.

```
#!/bin/sh
for filename in *; do
case $filename in
        *.c ) gcc -c "$filename" ;;
        *.cpp ) g++ -c "$filename";;
        *.txt) cat "$filename" ;;
        *) echo "$filename is not c or c++ or text code"

esac
done
exit 0
```

11.

1) d: directory , -: regular file, l:symbolic link
2) rwx for user, rx for group x for other
3) chmod o+x snap.doc
4) chmod g–r csc350

12.

- **gcc –c Fred.c**
  gcc –c Bill.c
  gcc –c foo.c
- **ar crv libBF.a bill.o fred.o**
- **mv libBF.a /home/separk/bin**
- **gcc –o foo /home/separk/bin foo.c -lBF**

13. .

- When any system call is called inside a process, since it is run on kernel's space, the process does not need use its own space (unbuffered).
- When any library function is called inside a process, since it runs on process's space, the process needs use its own space to save local variable for the library function (buffered).