1. Closed book. You do not need a calculator. You may use extra paper for scratch work or if you need more room for answers.
2. If you finish before 11:40 am, you may bring your papers to the instructor at the front of the room.
3. Budget your time carefully. Read over the entire exam before starting. There are 100 points in total.
4. Write legibly. If I cannot read it, I cannot give you credit for it.
5. Make sure your answer clearly indicates the result (versus your scratch work). Draw a box or a circle around your answers if needed.
6. Assume any code segment is embedded in a correct program.
7. When showing program output, you do not have to indicate the exact spacing but do show when an output starts on a new line.
8. If anything is ambiguous or unclear, ask the instructor.

I pledge that I have neither received nor given unauthorized aid on this examination.

Signed: ________________________________________________

PRINT: _________________________________________________

ID: ____________________________________________________
Part 1. Multiple Choice (15 points in total, 3 points for each):

Note: Each question may contain one or more answers.

1) In shell script, a function is defined as “start(){… …}“. Then to call this function we can choose which command in the following choices?
   (a) start
   (b) start()
   (c) function start()
   (d) use start

   KEY: A

2) In the following commands, which one can write the output both to the stdout(screen) and file.
   (a) ls | tee list.txt
   (b) ls > list.txt
   (c) ls | cat > list.txt
   (d) (ls > list.txt; cat list.txt)

   KEY: AD

3) In the following commands, which one will invoke a subshell:
   (a) (ls;pwd;date )
   (b) ./fib.sh (fib.sh is a shell script)
   (c) date &
   (d) ls;pwd;

   KEY: ABC

4) To inhibit variable substitution replacement when echoing a string, we can use
   (a) Single quotes
   (b) Double quotes
   (c) Grave accent
   (d) None of above

   KEY: A

5) The following statements are related to the return value of UNIX process. Which one is the true statement?
   (a) Every Unix process terminates with an exit value
   (b) The returned value 0 means failure for UNIX process.
   (c) All built-in commands return non-zero value when they fail
   (d) None of above
Part 2. Short answers. (20 points in total, 2 points for each):

1) List all .sh files in current directory.

   `ls * .sh`

2) Count the number of .sh files in current directory.

   `ls * .sh | wc -l`

3) Output the results of 2) in a message “There are **shell script files” where ** is the number.

   `echo there are `ls * .sh | wc -l` * shell script files`

4) Define an environmental variable named “course’ with initial value “csc3320”.

   `export course=csc3320`

   or
   
   `course=csc3320; export course`

5) Execute command “sleep 20” in background.

   `sleep 20 &`

6) List the all running processes in current system and redirect the output to currentPS.txt.

   `ps -e > currentps.txt`
7) Output the process ID belonging to application “safari”. (hint: the application has a command containing string “Safari”).

```
ps -e > grep “safari” >awk '{printf $1}''
```

8) Kill the process belonging to “Safari”.

```
ps -e > grep “safari” >awk '{printf $1}'>kill
```

9) What does the first line “#!/bin/ksh” mean in a shell script.

Let k-shell execute this script.

10) What is the purpose of the environmental variable $PATH?

   $PATH contains the locations/paths where system looks for commands

Part 3. Complete the scripts. (25 points in total):

1) The following shell script is used to copy directories under /home/ test/ to /home/backup/.

   (10 points)

```
#!/bin/bash

for dir in ____________; do
  if [ -d $dir ]; then
    ____________
  fi
done
```
2) The following shell script is used to check the input IP address is invalid or not. (15 points)

```bash
IsValidIP() {
    # use egrep to check if the input IP address follows the IP address pattern
    echo $1 | _____________________________________________
    match=$?
    if (( match==0 )); then
        a=`echo $1 | awk -F. '{print $1}'`
        b=`echo $1 | awk -F. '{print $2}'`
        c=`echo $1 | awk -F. '{print $3}'`
        d=`echo $1 | awk -F. '{print $4}'`
        for n in $a $b $c $d; do
            if ______________________________; then
                echo "the number of the IP should not be greater than 255 and less than 0"
                return 0
            fi
        done
    else
        echo "The IP format you input is wrong, the format should be like 192.168.100.1"
        return 1
    fi
}
# main program
rs=1
while [ $rs -gt 0 ]; do
    read -p "Please input the ip:" ip
    ______________________________ # call the function IsValidIP
    rs=$?
done
echo "The IP is valid!"
```
Part 4. Correct errors in the scripts. (15 points in total):

The following shell script is used to output the summation of all integer numbers between 1 and 100 that are divisible by 3.

```bash
#!/bin/bash
#sum test
sum = 0
for a in 1..100
    do
        if ((( $a % 3 != 0 )) then
            continue
        fi
    sum=$sum+$a
    done
echo ‘the sum is $sum’
```

Which lines are incorrect? And how to correct it? Please write down your solution for that line.

<table>
<thead>
<tr>
<th>Line Numbers</th>
<th>Correct statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>sum = 0 → sum=0</td>
</tr>
<tr>
<td>4</td>
<td>1..100 → {1..100}</td>
</tr>
<tr>
<td>6</td>
<td>if ($( $a % 3 != 0 )) then → if ($( $a % 3 != 0 )); then</td>
</tr>
<tr>
<td>9</td>
<td>sum=$((sum+$a)) → ((sum+=$sum+$a))</td>
</tr>
<tr>
<td>11</td>
<td>echo ‘the sum is $sum’ → echo “the sum is $sum”</td>
</tr>
</tbody>
</table>
Part 5. Writing a program (15 points)
Assuming there is an offline game, once a player finish the game, the final score will be stored in a score file. Now we have multiple players playing that game. Each player needs to play that game in two computers “Alpha” and “Beta”. So there are two score files generated separately in two computers. Our goal is to check those two files and join them to create a final file “score_final.txt” for future statics.

Suppose the score file in computer “Alpha” has been renamed as “score_alpha.txt”; in computer “Beta” the score file has been renamed as “score_beta.txt”.
In a score file, a colon separates the player’s name and score. One example of “score_alpha.txt”, “score_beta.txt” and final score file “score_final.txt” are as follows:

- **score_alpha.txt**

  Allen:12
  Bob:25
  Carl:36
  Kevin:121
  Tomy:99
  Jack:108

- **score_beta.txt**

  Jack:9
  Carl:8
  Kevin:11
  Tomy:22
  Bob:5
  Allen:88

- **score_final.txt**

  Allen:12:88
  Bob:25:5
  Carl:36:8
  Jack:108:9
  Kevin:121:11
To accomplish this task, we can use a utility `join` in UNIX, and the commands could be:

```
$ sort score_alpha.txt > score_alpha.tmp
$ sort score_beta.txt > score_beta.tmp
$ join -t ":" score_alpha.tmp score_beta.tmp > score_final.txt
$ rm score_alpha.tmp score_beta.tmp
```

Now you are asked to write your own `join` utility, named as “MyJoin”, to finish the same task. (You cannot use `join` in this program)

Please write down a bash shell script “MyJoin”.
The command format to use `MyJoin` could be

```
$ ./MyJoin score_alpha.txt score_beta.txt > score_final.txt
```

Key:
```
#!/bin/bash
# Author: Xuan Guo
awk –F: '{print $1}' $1 > score_r.tmp
awk –F: '{print $1} $2 >> score_r.tmp
sort score_r.tmp|uniq > score.tmp
for n in `cat score.tmp`
do
    echo -n ${n}:
    egrep -p "$n" $1
    exist=$?
    if ((( exist == 0 )))
    then
        egrep "$n" $1|awk –F: '{printf "%d:", $2}'
    fi
    egrep -p "$n" $2
    if (( exist==$? ))
    then
        egrep "$n" $2|awk –F: '{printf "%d", $2}'
    fi
do
rm score.tmp
```