CS 60
MID-TERM EXAM
April 27, 2010
WRITE ALL YOUR ANSWERS ON SPACE PROVIDED.
ANSWER ALL QUESTIONS. TOTAL POINTS ARE 100.
ONLY STUDENTS THAT ARE REGISTERED FOR THIS COURSE MAY TAKE THIS EXAM.

NAME:_____

1 { Circle for each part True or False depending whether or not the statement is true or false. Each question is worth 1.5 Points}

- { True or **False** } The function `malloc` returns a pointer to a block of memory all of which has been initialized to zero.
- { **True** or False } A structure (`struct`) in C may contain inside it a structure (`struct`) that contains inside it a structure (`struct`).
- { True or **False** } The command `pwd` is normally used to change your password.
- { **True** or False } The memory allocated through `calloc` is in the area of memory called a Heap.
- { True or **False** } Unix was created by researchers at IBM.
- { True or **False** } The command `cd teo` is used to jump to the directory of the user `teo`.
- { True or **False** } There is no such thing as a pointer to a structure (`struct`) in C.
- { True or **False** } In Linux when we are editing a file via `vim` or `vi` and we press (at the same time the two keys) `ctrl z` it takes you back to command mode after saving the file you are editing and then killing the `vim` or `vi` process (program).
- { True or **False** } In C (`printf`) one uses `%d` to print `int` numbers in hexadecimal.
- { True or **False** } In C when we are allowed to pass a `struct` to a function, but the function cannot return a `struct`.

2 {Simple Questions}

a.- [4 points] Suppose that the declaration `int x[4] = {6,7,8,9}` is in a C++ program. What is the value of `x[2]` just after the above declaration is executed? Is `x[5]` initialized to zero?

b.- [4 points] What is the value of `s` after executing the following code?

```
#include <stdio.h>
int main(void)
{  int i,s;
   s = 2;
   i = 3;
   {int i;
    for ( i = 1; i<= 10; i++, s++)   Ans:   3   6
       { if (i > 4) break;
             else continue;
       }
    }
   printf("%d %d\n", i,s);
}
```

c.- [6 points] What does the following C++ code print in the standard output?

```
#include <stdio.h>
struct node {
  int  data;
  struct node* link;
};

int main(void)
{
  struct node y;
  struct node z;
  y.data = 3;
  z.data = 2;
  y.link = &z;                    ANS: 8 5
  z.link = &y;
  (y.link)->data = y.data + z.data;
  (z.link)->data = z.data + (z.link)->data;
  printf("%d %d\n", y.data, z.data);
}
```

e.- [8 points] In the directory entry

```
-rwxr-xr-x 1 abc student 56056 May 11 11:11 test.txt
```

- Who is the Owner of the file? `ANS: abc`
- What is the Group of the file? `ANS: student`
- What is the name of the file? `ANS: test.txt`
- What is the size of the file in bytes? `ANS: 56056`
- What permissions does the Owner have on the file? `ANS: rwx`
- What permissions does the Group have on the file? `ANS: rx`
- What permissions does every one else have on the file? `ANS: rx`
- When was the file last modified? `ANS: May 11 11:11`

f.- [3 points] If the above file test.txt is the only file in the current directory how do you display the above information about the file?

`ls -l (or ll)`

g.- [3 points] Briefly explain what does `chmod 751 test.txt` do to the above directory entry?

`Changes -rwxr-xr-x to -rwxr-x--x`

pace*0.25in j.- [3 points] Suppose there is a program called prog1 that reads from the standard input. If you want to redirect the file input.txt to the standard input of prog1, what command would you type?

`./prog1 < input.txt`

3 {Makefile}

[8 points] Given the following make file, and directory contents give the exact commands that are executed when make is run.

```
Directory Contents:

-rw-r--r--  1 cs60 guest  115 Apr 23 14:31 helper1.c
-rw-r--r--  1 cs60 guest   75 Apr 24 14:15 helper1.h
-rw-r--r--  1 cs60 guest 1734 Apr 23 14:33 helper1.o
-rw-r--r--  1 cs60 guest 1404 Apr 23 14:26 helper2.c
-rw-r--r--  1 cs60 guest  254 Apr 22 14:48 helper2.h
-rw-r--r--  1 cs60 guest 2342 Apr 23 14:33 helper2.o
-rw-r--r--  1 cs60 guest 1178 Apr 23 14:31 main.c
-rw-r--r--  1 cs60 guest 1139 Apr 23 14:33 main.o
-rwxr-xr-x  1 cs60 guest 4325 Apr 23 14:34 main*
-rw-r--r--  1 cs60 guest  505 Apr 23 14:00 Makefile

Makefile:

main: main.o helper1.o helper2.o
      gcc -Wall -g -o main main.o helper1.o helper2.o  <----------
main.o: main.c helper1.h helper2.h
        gcc -Wall -g -c helper1.c helper2.c main.c  <----------
helper1.o: helper1.c helper2.c main.c helper1.h
        gcc -Wall -g -c helper1.c helper2.c main.c  <----------
helper2.o: helper1.c helper2.c main.c helper2.h
        gcc -Wall -g -c helper1.c helper2.c main.c
```

4 {Questions}

(a) [6 points] What does the following code print?

```
#include <stdio.h>
int main(void)
{
   int x = 5;
   int y = 20;
   int *pi = &y;
   int *ri = &x;
   int *qi;
   qi = ri;
   ri = pi;
   pi = qi;
   (*pi) = y;
   (*ri) = x + y;       ANS:  20   20    40
   (*qi) = (*pi);
   pi = &x;
   printf("%d %d %d\n", (*pi), (*qi), (*ri));
}
```

(b) [4 points] What does the following code print?

```
#include <stdio.h>
int main(void)
{
        int i=1;
        int j=3;
        {
                int i=3;
                j--;
                i = i+j;                ANS:  6  2
                i++;                          1  3
                printf("%d %d\n", i, j);
                j++;
        }
        printf("%d %d\n", i, j);
}
```

c.- [4 points] Clearly indicate the value printed by the following C code.

```
#include <stdio.h>
```

```
void swap(int z, int y)
{
  int temp;
  int *pz;
  int *py;
  pz = &z;
  py = &y;
  temp = *pz;
  *pz = *py;
  *py = temp;                  ANS:   10  20
}
int main(void)
{
  int x = 10;
  int y = 20;
  swap(y, x);
  printf("The value of x is %d, and y is %d\n",x,y);
}
```

5 {Questions}

a.- [6 points] For the code given below clearly indicate what the `print` line prints.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int i;
  int *parr;
  parr = (int *) calloc(5,sizeof(int));
  for (i=0; i<5; i++)
      *(parr+i) = i;              ANS:  3  1  2
  parr[0] += 3;                         4 4 2 3
  printf("%d %d %d\n",*(parr),*(parr+1),*(parr+2));
  *(parr+1) += *parr;
  printf("%d %d %d %d\n",parr[0], parr[1], parr[2], parr[3]);
  free(parr);
}
```

b.- [4 points] For the code given below clearly indicate what the `print` line prints.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  struct stuff {
    char *a;
    int *b;
    double *c;                              ANS: 16
    long double *d;                              16
    } var;                                       16
  int num = sizeof(var);                          4
  printf("%d\n",num);
  var.b = (int *) malloc(5*sizeof(int));
  num = sizeof(var);
  printf("%d\n",num);
  var.c = (double *) malloc(5*sizeof(double));
  num = sizeof(struct stuff);
  printf("%d\n",num);
  var.d = (long double *) calloc(5,sizeof(long double));
  num = sizeof(var.d);
  printf("%d\n",num);
}
```

(c) [4 points] What does the following C code print when executed?

```
#include <stdio.h>
int main(void)
{
int num = 0;
int val;
for (val=0; val < 3; val++)
  {
   switch (val)
     {
       case 0 : num += 9;break;     ANS:  9 0
       case 1 : val += 3;                 12 3
       case 2 : num += 2; num++; val--;
     };
   printf("%d %d\n",num,val);
  }
}
```

(d) [4 points] This C code is intended to initialize an array of values. Correct any errors or state that there are none:

```
int i, j, arr[5][8];
for (i=1; i<5; i++)     --> for (i=0; i<5; i++)
  for (j=1, j<8, j++)   --> for (j=0; j<8; j++)
    arr[i][j] = 0;
```

6 {Code}

[15 points] Write a simple C program to read in 20 positive integer numbers (one per line). The program will then count and print the number of elements larger than the average value of the 20 elements.

7 {Homework 3}

[10 Points] Suppose that instead of 4 piles for homework 3 we have 5 piles. A student claims that the game will always be won no matter what is the ordering of the cards. Prove or disprove the student's claim. To prove it, give arguments to convince the grader that the game will always be won (no matter what the ordering of the cards is). To disprove it, give an ordering of the input cards for which the program will lose.

The game will always be won. The reason for this that after dealing a card and making all the possible moves, one will be left with at most four piles with one exposed card each and at least one pile empty. We call this situation condition C. Initially condition C holds. Assume that condition C holds after dealing i cards, we now show that condition C will hold after dealing i+1 cards (and making all possible move operations). Lets deal the i_1st card. Now we will discard zero or more cards. At this point we know that at most four piles will have cards. If one pile has two cards, one of the cards will be moved to an empty pile. Now we apply discarding zero or more times. At this point we know that at most four piles will have cards and no pile will contain two or more cards. So condition C will hold. By induction and since aces will never be discarded, it will be the case that at the end we will have the four aces exposed and the game will always be won.