8 pts. 1. Which of the following statements about *exit-condition* loops is false?

(a) One iteration through the loop is always performed.
(b) They are less commonly used than entry-condition loops.
[✓ (c)] The test for loop continuation is conducted prior to entering the loop.
(d) In C, they can be implemented using the do-while loop structure.

10 pts. 2. Explain what the following function does.

```
int func(int num) {
   return num < 0 ?  -num :  num;
}
```

Returns the absolute value of num.

If num is less than 0, the expression returns -num; otherwise it returns num. This is the definition of absolute value.

10 pts. 3. The following program should print “x is odd”, but always prints “x is even” instead, no matter what value is passed to the oddOrEven() function. Make the correction that will cause it to work correctly. Hint: this is one of the most common errors in C programming.

```
int main()
{
   oddOrEven(3);
}

void oddOrEven(int num)
{
   int remainder = num % 2;
   if (remainder = 0)              ==
      printf("x is even\n");
   else
      printf("x is odd\n");
}
```

12 pts. 4. Given the following program:

```
int main() {
   int i=2, result=0;
   while (i <= 20) {
      result += i;
      i++;
   }    return result;
}
```

Fill in the missing details of the for loop in order to make the program below equivalent (in terms of what it computes) to the program above:

```
int main() {
   int ~~i, result;~~
   for ( i= 2, result=0;  i <= 20  ;  i++          )
      result += i;
   return result;
}
```

10 pts. 5. **What does the following program print:**

```
main() {
   int num=12;
   switch(num) {
      case 1:      num += 10;
      case 3:      num -= 8;
      case 2:      num--;
      case 0:      num++;
                   break;
      default:     num = -1;
   }
   printf("num = %d\n", num);
}
```

num = -1

since num is initialized to 12, the default case is taken, where num is set to -1;

10 pts. 6. **What does the following program print:**

```
main() {
   int num=2;
   switch(num) {
      case 1:      num += 10;
      case 3:      num -= 8;
      case 2:      num--;
      case 0:      num++;
                   break;
      default:     num = -1;
   }
   printf("num = %d\n", num);
}
```

num = 2

since num is initialized to 2, case 2: is taken, where num is decremented to 1. "Fall-through" then occurs and num is incremented back to 2.

10 pts. **7. What does the following program print:**

```
main() {
   int a = 10;
   if (a%2 == 0) {
      printf("%d is divisible by 2\n", a);
      if (a%4 == 0)
         printf(", %d is also divisible by 4\n", a);
   }
   else
      printf("%d is odd\n", a);
}
```

since a = 10, a % 2 = 0, and the if condition is satisfied, causing the first printf to occur. But a % 4 = 2, so the second if condition is not met, and nothing else is printed.

$\boxed{\text{10 is divisible by 2}}$

10 pts. **8. What does the following program print?**

```
main() {
   int i, result=0, a[4] = {1,2,3,4};
   for(i=0; i<4; i++) {
      result += a[i];
   }
   printf("result = %d\n", result);
}
```

i	a[i]	result
		0
0	1	1
1	2	3
2	3	6
3	4	10

$\boxed{\text{result = 10}}$

10 pts. **9. What does the following program print?**

```
main() {
   int i, result=0, a[6] = {1,2,3,4};
   for(i=3; i>=0; i--) {
      result += a[i];
   }
   printf("result = %d\n", result);
}
```

i	a[i]	result
		0
3	4	4
2	3	7
1	2	9
0	1	10

$\boxed{\text{result = 10}}$

10 pts. **10. What does the following program print?**

```
main() {
   int i, result=0, a[6] = {1,2,3,4};
   for(i=0; i<4; i++) {
      if (i == 2)
         break;
      result += a[i];
   }
   printf("result = %d\n", result);
}
```

i	a[i]	result
		0
0	1	1
1	2	3
2	* break	

result = 3