ICP Midterm

Answer each question briefly.
24 questions in total $-\ 4\% \times 22 + 6\% \times 2$. **** Don't rush – there is plenty of time.**

1 a) The following loop is meant to print out all the unsigned short integers.
But, it causes a runtime error. Figure out the error.

```
for (unsigned short i=0;i<=USHRT_MAX;i++) printf("%hu ",i);
```

b) What can you say about the following loop that is meant to print out all the signed short integers?

```
for (short i=SHRT_MIN;i<=SHRT_MAX;i++) printf("%hd ",i);
```

Hint: The two loops in a) and b) don't necessarily have the same destiny.

2 Show the output of each **printf** below.

a)
```
int x=2,y=3;                          // the input is NT$ 23,  456 ↵
printf("%d\n",scanf ("NT$%d,%d.00\n",&x,&y));
printf("%d\n",printf("NT$%d,%d.00\n",x,y));
```

b)
```
signed short z=-2;
printf("%ho %x",z,z);          // Hint: Beware of %x – it isn't %hx.
```

c)
```
unsigned char c=2626;
printf("%d %c",c);
```

Hint: Convert 2626 to unsigned char.

Hint: The ASCII codes of the characters '0', 'A', and 'a' are 48, 65, and 97, respectively.

3 Consider the following two **C**++ expressions

a) `1==1? -1: 1u`

b) `2*('1'>'0')+2.0f*true`

Answer the following questions for each expression.
What is the value of the expression? What is the type of the value?
How many type conversions are necessary to evaluate the expression?

4 Fill in the following blanks

a)
```
bool find(int k,int a[],int n) // check if k ∈ a[0..n − 1]
{
    int i;
    for (i=0;______;i++);
    return i<n;
}
```

b)
```
int max(int a[],int n)  // find the maximum element in a[0..n-1]
{
    int m=a[0];
    for (int i=1;i<n;i++)
       m = ______;          // Hint: Conditional expression
    return m;
}
```

5 Show the output of each loop below.

a)
```
int i=0,j=0;
while (++j<=5) printf("%d",i+++j);
```

b)
```
for (int i=1;i<=9;i+=i%2==0?3:1)
    for (int j=i;j>=1;j-=2)
       printf("%d",j);
```

6 The following function is meant to compute the product of **m** and **n** in case of no overflow, and return 0, otherwise.

```
unsigned product(unsigned m,unsigned n)
{
    return m*n<=UINT_MAX? m*n: 0u;
}
```

a) Explain why the function fails to perform the task.

b) Correct it. **Hint**: Beware of division by zero.

7 a) Show the IEEE 754 representation of the single-precision floating-point number **3.7f** in hexadecimal notation. Show your work. (6%)

b) Write down a chunk of code to print out the representation of **3.7f** in hexadecimal notation.

8 Given an integer $n \geq 0$, each incomplete loop below is meant to set the variable **ss** to the value of the expression $1 + (1 + 2) + (1 + 2 + 3) + \cdots + (1 + 2 + \cdots + n)$ in a different way. Fill in the blanks to complete the loops.

a) // This loop, taken from HW2, uses $1 + 2 + \cdots + (i - 1)$ to compute $1 + 2 + \cdots + i$.
```
int s=0,ss=0;
for (int i=1;i<=n;i++) _____;
```

b) // Since the summands include n 1's, $n - 1$ 2's, $n - 2$ 3's, and so on, this loop
// computes $1 \times n + 2 \times (n - 1) + 3 \times (n - 2) + \cdots$.
```
int ss=0;
for (int i=1;i<=n;i++) _____;
```

9 a) Consider the following function from HW3

```
bool palindrome(unsigned n)      // check if n is a palindrome
{
    unsigned r=1;
    while (n/r>=10) r*=10;
    while (____) {                // termination condition omitted
        if (n/r==n%10) { n%=r; n/=10; r/=100; }
        else return false;
    }
    return true;
}
```

Which *can* be used as the termination condition of the while loop? **Hint:** Multiple choices

① `r>=10` ② `r>0` ③ `n>=10` ④ `n>0`

b) The following incomplete function is meant to check if n is a palindrome by extracting its digits into an array and then comparing digits from both ends of the array successively.

```
bool palindrome(unsigned n)
{
    unsigned char d[10];
    int i=0;
    while (n>=10) { d[i]=n%10; n/=10; i++; }  // digit extraction
    d[i]=n;
    // compare digits from both ends of the array d successively
}
```

Complete the function by supplying the missing code. (6%)

10 The following code, taken from the function **bad** of HW4,

```
unsigned char d[10]={0};
while (n>=10) { d[n%10]++; n/=10; }  //*
d[n]++;                              //*
```

sets

`d[`k`]` = the number of times the digit k occurs in the unsigned integer `n`, $0 \le k \le 9$.

a) Explain why the two starred lines *cannot* be replaced by the loop

```
while (n>0) { d[n%10]++; n/=10; }
```

b) On the other hand, the two starred lines *can* be replaced by the loop

```
do { d[n%10]++; n/=10; } while (n>0);
```

Howover, this is less efficient. Why?

Hint: Count the number of divisions, including `%` and `/`.

11 Consider the function **good** of HW4

```
bool good(unsigned n)
{
    unsigned short a[8];
    int i=0;
    while (n>=1000) { a[i]=n%1000; n/=10; i++; }
    a[i]=n;
    for (int j=i;j>=1;j--)                  //*
        for (int k=j-1;k>=0;k--)            //*
            if (a[j]==a[k]) return false;   //*
    return true;                            //*
}
```

a) Rewrite the starred lines so as to use **break** statements to exit the nested loops.
Hint: Need two **break** statements – one for each loop.
Also, beware of the scopes of the control variables **j** and **k**.

b) Rewrite the starred lines so that *neither* **return** *nor* **break** is used to exit the nested loops.
Hint: Use a boolean variable.

12 Consider the math function

$f : \mathrm{Z} \times \mathrm{N} \to \mathrm{Z}$

defined by

$f(m,n) = (m + \lfloor\sqrt{n}\rfloor)^2 + 3(m + \lfloor\sqrt{n}\rfloor) + 5$

where Z and N are the set of integers and nonnegative integers, respectively.
Being an *expert* in C programming, you are asked to define this function in C.
Note: Substitute **int** for Z and **unsigned** for N
Hint: As an expert, be efficient.