ICP Final

1 Given

```
int a[5]={1,2,3,4,5};
int *p=a,*q=a+3;
```

Show the output of each independet part below. (6%)

a) `printf("%d",p-q);`

b) `printf("%d",q[-1]);`

c) `printf("%d",*p++);`

2 What is wrong with each part below? (6%)

a) `*("Smoopy"+1)='n';`

b) `*(int*)malloc(sizeof(int))=777;`

c) `int* f(int x) { return &x; }`

3 Given

```
int (*(*c)[2])[3]
```

a) Draw a picture showing the simplest structure that can be bound to variable **c**. (4%)

b) What is the type of each expression below? (4%)

1) `*c`

2) `&c`

4 Show the output of each program below. (8%)

a)
```
#include <stdio.h>
int f(int x) { return x+1; }
#define f(n) ((n)==0? 1: (n)*f((n)-1))
int main() { printf("%d\n",f(5)); }
```

b)
```
#include <stdio.h>
void p(int n)
{
   printf("%d ",n);
   if (n>0) p(-n); else if (n<0) p(-n-1);
   printf("%d ",n);
}
int main() { p(2); }
```

5 Fill in the following blanks.

a) This function sorts the array **a**$[0..n-1]$ by bubble sort that *extends the sorted subarray as far as possible on each pass*. (4%)

```
void bsort(int* a,int n)
{
    for (int i=0;i<n-1;) {
        int k= (1) ;
        for (int j=n-1;j>i;j--)
            if (a[j-1]>a[j]) {
                int z=a[j]; a[j]=a[j-1]; a[j-1]=z; k=j;
            }
        (2) ;
    }
}
```

b) This function sorts the array **a**$[0..n-1]$ by insertion sort that uses binary search to locate the insertion point. (4%)

```
void isort(int* a,int n)
{
    for (int i=1;i<=n-1;i++) {
        int l=0,h=i-1;
        while (l<=h) {
            int m=(l+h)/2; if (a[i]<a[m]) (3) ; else (4) ;
        }
        int key=a[i];
        for (int j=i-1;j>=l;j--) a[j+1]=a[j];
        a[l]=key;
}
```

6 a) Given

```
struct stack { int top,stk[80]; }
```

and assume that a stack **s** is empty iff **s.top==-1**.

The following function computes the number of elements contained in stack **s**:

```
size_t size(stack s) { return s.top+1; }
```

What is the functionality of the parameter **s** of function **empty**? (2%)

What is the drawback of the parameter passing method used for the parameter **s**? (2%)

b) Rewrite the function **size** with a better parameter-passing method so that it retains the functionality of the parameter **s** and doesn't have the drawback of part a). (4%)

7 The following code generates 5 random numbers in increasing order

```
int a[5];
loop: for (int i=0;i<5;i++) a[i]=rand();
for (int i=0;i<4;i++)
    if (a[i]>=a[i+1]) goto loop;
```

Rewrite the code without using **goto** statement. (6%)

8 Consider the following program

```
void p(int n,int* r)
{
    static int x=n;
    if (n==0) *r=x; else p(n-1,r);
}
int main(void)
{
    int* r=(int*)malloc(sizeof(int));
    p(2,r); free(r);
}
```

Draw a picture showing the contents of the global/static data area, the runtime stack, and the heap at the point when the statement ***r=x** is executed. (6%)

9 Given (12%)

```
struct node { int datum; node* succ; } *head;
```

and consider the following linked lists in which the successor of the last node is the node pointed to by **head**:

(1) **head** → 8 ↺ (2) **head** → 7 → 8 ↺

a) Write a piece of code to insert a node containing 7 into the list (1) to obtain the list (2).

b) Write a piece of code to delete the node containing 7 from the list (2) to obtain the list (1).

10 Let **Happy.exe** be the executable of a C program whose function **main** is declared as

```
int main(int,char **argv) // the 1st parameter is unnamed
{
    int argc;
    // missing code
    printf("%d\n",argc);
}
```

a) The function **main** does NOT rely on the 1st parameter to receive the number of strings appeared in the command line. Instead, it counts that number by itself with the help of the 2nd parameter **argv**. Write down the missing code to accomplish the mission. (4%)

b) Suppose the following command is entered within a command interpreter:

```
prompt> Happy New Year!
3
```

Draw a picture showing the data structure bound to **argv**. (4%)

11 a) The library function **strlen** computes the number of characters, excluding **'\0'**, in a string. Which is the correct signature of **strlen** and why? (4%)

1) `size_t strlen(char*);`
2) `size_t strlen(const char*);`

b) Give a *recursive* function of the function **strlen**. (4%)
Note: This part is independent of part a), i.e. you may use any signature given in part a.

12 [Inspired by HW5 on BCD]

a) Given `bcd b=dec2bcd(123);`
write an expression to extract the digit 3 from the bcd number **b**. (4%)

b) Show the output of the following statement.
`printf("%d",0x123-bcd2dec(0x123));` (4%)

13 [Inspired by HW7 on Coin Change]

Let **a**$[0..n-1]$ be an array that contains n integers of the form

$\underbrace{11\ldots11}_{x\ 1's}\underbrace{22\ldots22}_{y\ 2's}\underbrace{33\ldots33}_{z\ 3's}$ where $x+y+z=n, x\geq 0, y\geq 0, z\geq 0$

Given such an array, the following function computes x, y, and z and print them out.

```
void show(int* a,int n)
{
    for (int k=1;k<=3;k++) {
        int c=0;
        for (int i=0;i<n;i++) if (a[i]==k) c++;
        printf("%d ",c);
    }
}
```

a) Albeit correct, it is inefficient. Why? (4%)

b) How would you modify the code to make it efficient? (4%)