ICP Final solution

1 a) -3

b) 3

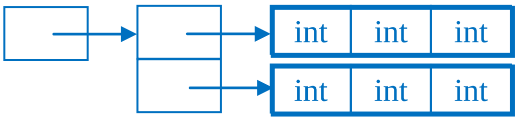
c) 1

2 a) `"Smoopy"`is a constant and can’t be modified.

b) See Practice D1b

c) The call `f()` yields a dangling pointer, since the storage for `x` no longer exists after returning from the function `f`.

3 a)



b) 1) `int (*[2])[3]`

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

4 a) See Practice E6

b) -2 1 -1 0 0 -1 1 -2 2

5 a) (1) `n-1` or, any value ≥ $n - 1$

(2) `i=k`

b) (3) `h=m-1`

(4) `l=m+1`

6 a) functionality – in

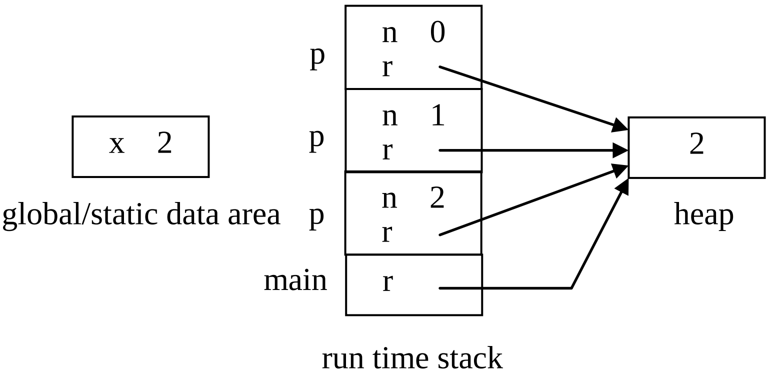
drawback – call-by-value takes time and space to copy the stack

b)

```
size_t size(const stack* s)// or, const stack*const s
{
    return s->top+1;
}
```

7
```
while (true) {
    for (int i=0;i<5;i++) a[i]=rand();
    int i;
    for (i=0;i<4;i++)
        if (a[i]>=a[i+1]) break;
    if (i==4) break;
```

8

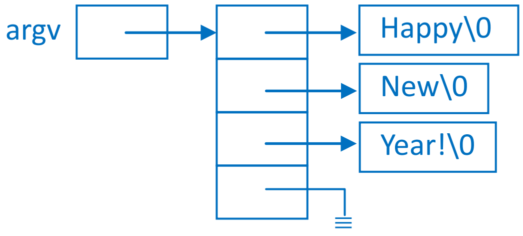


9 a)
```
node* p=(node*)malloc(sizeof(node));
p->datum=7;
p->succ=head;
head->succ=p;  // or, p->succ->succ=p;
head=p;
```

b)
```
node* p=head;
head=head->succ;
head->succ=head;
free(p);
```

10 a)
```
argc=0; while (*argv++) argc++;
```

b)



11 a) See Practice P14a)

b)

```
size_t strlen(const char* s)
{
    return *s? 1+strlen(s+1): 0;
}
```

12 a) `b&0xf`

b) 168

13 a) The array is scanned three times.

The code in part b) scans the array only once.

b)

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