1. Suppose the following code is executed on a 32-bit machine, where “**int**” is 4 bytes, “**short**” is 2 bytes, “**char**” is 1 byte and “**pointer**” is 4 bytes.

```
struct data {
    char a;
    short b[3];
    char* c;
    union
    {
        short   s;
        int     i;
        char    c;
    }p;
    char d;
};
structdata d[2];
```

Suppose the address of global variable dis 0x8049600, please answer the following questions.

Variable	start address
d[0]	0x8049600
d[1]	0x8049614
d[0].a	0x8049600
d[0].b[2]	0x8049606
d[0].c	0x8049608
d[0].p.c	0x804960c
d[0].p.i	0x804960c
d[0].d	0x8049610

2. In the following code, A and B are constants defined with #define:

```
typedef struct{
short x[A][B]; /*Unknown constants A and B*/
  int y;
}str1;

typedef struct{
  char array[B];
  int t;
  short s[B];
  int u;
}str2;

void setVal(str1 *p, str2 *q){
  int v1 = q->t;
  int v2 = q->u;
  p->y = v1 + v2;
}
```

Gcc generates the following code for the body of setVal:

```
1 movl 12(%ebp), %eax
2 movl 36(%eax), %edx
3 addl 12(%eax), %edx
4 movl 8(%ebp), %eax
5 movl %edx, 20(%eax)
```

What are the values of A and B?

A:1 B:9

A:1 B:9

III. Writing out the following definitions, you can use the variable c. Examples:

An integer : int c;
A pointer to an integer : int *c;

1) A pointer to a pointer to an integer: int **c;
2) An array of 7 pointers to integers: int *a[7];
3) A pointer to an array of 7 integers: int (*a)[7];
4) A pointer to a function that takes an integer argument and returns an integer: int (*c)(int);
5) An array of 7 pointers to functions that take an integer argument and return a pointer to an integer: int * (*a[7](int));