# DSA I: LAB ASSIGNMENT 5
# SY(COMP)

---

For all the following problems, you have to submit JPEG images. You can draw images using any tool on Linux/Windows and submit. xfig or draw.io are tools for same. You can also use Libreoffice Draw or GIMP.
xfig is a tool for drawing images. The interface of xfig is slightely different, learn it. Save images from xfig in .fig format and then save in .jpg format (this can be done by using file->export->language option). You have to submit images in both formats. (xfig saves the images using vector formats like .fig - these images can be stretched to any extent without loosing resolution (blurring)) .
In the diagram only show final data structure. No need to draw intermediate data structure. Do show value for very variable set or used in the code. (e.g. int, i,j, x, y, etc.)
Draw diagrams.

**1)**

```
typedef struct entry {
    double x;
    struct entry *e;
}entry;

entry *f(entry **epp) {
    entry *ep = *epp;
    ep = (entry *)malloc(sizeof(entry));
    ep->e = NULL;
    ep->x = 1.2;
    return ep;
}
int main() {
    entry e, *ep, *eq;
    ep = &e;
    e.e = (entry *)malloc(sizeof(entry));
    eq = f(&(e.e));
    e.x = eq->x;
}
```

**2)**

```
typedef struct entry {
    double x;
    struct entry *e;
}entry;

entry *f(entry **epp) {
    entry *ep = *epp;
    *epp = (entry *)malloc(sizeof(entry));
    ep->e = NULL;
    ep->x = 1.2;
    return *epp;
```

```
}
int main() {
    entry e, *ep, *eq;
    ep = &e;
    e.e = (entry *)malloc(sizeof(entry));
    eq = f(&(e.e));
    e.x = eq->x;
}
```

**3)**

```
typedef struct entry {
    double x;
    struct entry *e;
}entry;

typedef entry *seq;
void f(seq *p) {
    int i;
    seq r;
    *p = (seq) malloc(sizeof(entry));
    r = *p;
    r->x = 100.0;
    for(i = 0; i < 2; i++) {
        r = (seq) malloc(sizeof(entry));
        (*p)->e = r;
        r->e = NULL:
        r->x = (double) i;
        (*p) = (*p)->e;
    }
}
int main() {
    entry *r;
    f(&r);
    r = r->e;
}
```

**4)**

```
typedef struct test {
    struct test **a;
}test;

int main() {
    test *m;
    int i;
    m = (test *)malloc(sizeof(test));
    m->a = (test **)malloc(sizeof(test *) * 3);
    for(i = 0; i < 2; i++) {
        m->a[i] = (test *)malloc(sizeof(test));
    }
    m->a[2] = (test*)malloc(3 * sizeof(test));
```

```
    m = f(&(m->a));
}
test *f(test ***ppp) {
    test *p, **pp;
    pp = *ppp; p = *pp;
    *pp = (test *)malloc(2 * sizeof(test));
    return pp[1];
}
```

**5)**

```
// For this problem assume user entres following data:
//  abhi  pune 1234
//  you  mum 987
//  jo  del 777134

typedef struct data {
    char name[8];
    char *address;
    unsigned long mobile;
    struct data **p;
}data;
int main() {
    data d; int k;
    char *greeting = "hi";
    char address[8];
    while(scanf("%s%s%ul", &d.name, address, &d.mobile) != -1) {
        d.address = malloc(strlen(address) + 1);
        strcpy(d.address, address);
        strcat(d.name, " ");
        strcat(d.name, greeting);
        k = d.mobile % 4;
        d.p = (data **)malloc(sizeof(data *) * k);
        for(i = 1; i < k + 1; i++) {
            d.p[i - 1] = (data *)malloc(sizeof(data) * i);
            d.p[i - 1][0] = d;
            memcpy(d.p[i - 1], &d, sizeof(d));
        }
    }
}
```

**6)**

```
typedef struct node {
    char c;
    struct node *next;
}node;
typedef node *list;
void append(list *l, int i) {
    list t;
    t = (node *)malloc(sizeof(node));
```

```
    t->c = i + 'a' + 2;
    t->next = NULL;
    if(*l == NULL) {
        *l = t;
        return;
    }
    (*l)->next = t;
    return;
}
int main() {
    int i;
    list  p;
    p = NULL;
    for(i = 0; i < 3; i++) {
        append(&p, i);
    }
}
```

**7)**

```
typedef struct node {
    int m;
    struct node *left, *right;
}node;
typedef node *tree;
void insert(tree *r, int i) {
    tree t, p, q;
    int j;
    t = (node *)malloc(sizeof(node));
    t->m = i;
    t->left = t->right = NULL;
    if(*r == NULL) {
        *r = t;
        return;
    }
    q = p = *r;
    j = i;
    while(i) {
        if(!p) q = p;
        if(i % 2 && p)
            p = p->left;
        else if(p)
            p = p->right;
        i--;
    }
    if(j % 2)
        q->left = t;
    else
        q->right = t;
    return;
}
int main() {
    int i;
```

```
    tree  p;
    p = NULL;
    for(i = 0; i < 3; i++) {
        insert(&p, i);
    }
}
```