```
//Q2
//A

void print_digets(int a , int b) {
	int len_b = 1;
	int tmp=10;
	while(b/tmp) {
	len_b++;
	tmp*=10;
	}
	tmp=1;
	for(int i=0;i<len_b-a;i++) tmp*=10;

	printf("%d" , b/tmp);
}

//B

int last_location(char* str , char ch) {
	int i = 0, ret = -1;
	while(*str) {
		if(*str == ch)
			ret = i;
		i++;
		str++
	}
	return ret;
}

//C

int count_space(char* str) {
	int count = 0;
	while(*str) {
		if(*str == ' ') count ++;
		str++;
	}
	return count;
}

char* add_word(char** pstr) {
	int wlen = 0;
	char* ref = *pstr;
	while(*ref != ' ' && *ref != 0){
		wlen++;
		ref++;
	}

	char* word = (char*)malloc(sizeof(char)*wlen +1);
	if (!word) return null;

	for(int i=0 ; i<wlen ; i++) {
		word[i] = ref[i];
	}
	//move the original pointer wlen + 1 chars forward
	*pstr=*pstr + wlen + 1;

	return word;
}

void cleanup(char** arr, int size){
	for(int i = 0; i<size;i++) {
		if(arr[i] != null) {
			free(arr[i]);
			continue;
```

```
            }
            break;
        }
        free(arr);
        return;
}

char** word_break(chat* str) {
        int spaces = count_space(str);
        char** arr = (char**)malloc(sizeof(char*) * spaces + 1);
        if(!arr) return null;

        int i = 0;
        while(*str) {
            arr[i] = add_word(&str);
            if(!arr[i]) cleanup(arr , spaces);
            i++;
        }

        return arr;
}




//Q3

typedef struct _node {
        int m[2][3];
        struct _node* next;
} Node, *pNode;

pNode creat(int mat[2][3]) {
        pNode n = (pNode)malloc(sizeof(Node));
        if(!n) return null;
        n->next = null;
        for(int i = 0 ; i<6 ; i++) n->m[i] = mat[i];

        return n;
}

void add(Node** head, pNode node) {
        Node** p = head;
        while(*p){
            p = &((*p)->next);
        }
        *p = node;
}

void print_node(pNode n){
        printf("%d %d %d \n", n->m[0],n->m[1],n->m[2]);
        printf("%d %d %d \n\n\n", n->m[3],n->m[4],n->m[5]);
}

void print(pNode head) {
        while(head){
            print_node(*head);
            head = head->next;
        }
        return;
}

void delete(Node** head) {
        Node** p = head;
        *head = null;
```

```
        while(*p){
                Node* tmp = *p;
                p = &((*p)->next);
                free(tmp);
        }
        return;
}

int main() {
        int m1[2][3] = {{1,2,3},{4,5,6}};
        int m2[2][3] = {0};
        int m3[2][3] = {1};

        Node* head = null;
        add(&head,creat(m1));
        add(&head,creat(m2));
        add(&head,creat(m3));

        print(head);
        delete(&head);


}
```