Description

printf("%d", n);

FORMAT ARGUMENT TYPE

%d, %i int decimal

%u unsigned int

%x unsigned int hexadecimal

%#x unsigned int hexadecimal with prefix 0x

%f double

%Lf long double

%e, %E double scientific notation

%c int to print a character

%s char * string (char array ended with '')

%p void * print memory address

%g, %G double %f or %e depending on the length

scanf("%d", &n);

FORMAT ARGUMENT TYPE

%d int * &n, store the input integer in n

%ld long *

%lld long long *

%u unsigned int *

%f float * read float

%lf double * read double

%Lf long double * read long double

%c char * read 3 characters %3c

%s char * read a string until whitespace

%n int * with %s, to get string length

char a[100]; int len;

scanf("%s%n", a, &len);

String and character functions:

#include

strcmp(str1, str2), strcpy(dest, src), strncmp(str1, str2), strncpy(dest, src)

Note that if str1 and str2 are the same, the return value of strcmp(str1,str2) is 0.

#include

isspace(ch), islower(ch), isupper(ch), isdigit(ch)

isalpha(ch), toupper(ch), tolower(ch)

How to avoid common errors for OJ:

1. Put the arrays in the 'global' area. Set their size bigger than required. Avoid using variable-length arrays (e.g. int arr[n];). Keep the array size fix (e.g., int arr[10000];).

2. After writing the code for reading input data, you may print out the data to check if your code reads them correctly. Do not proceed to write subsequent code before you confirm that.

3. If your program crashes, usually it is caused by memory related errors. Check the ranges of for-loops to see if your code attempts to read or write the elements out of the arrays' boundary.

Type definition and functions for linked lists:

```
typedef struct _node {
   int id;
   struct _node *next;
} Node;

void insertNode(Node* p, int id)
{
     Node *q = (Node *) malloc(sizeof(Node));
     q->id = id;
     q->next = p->next;
     p->next = q;
}
Node* deleteNode(Node* p)
{
     Node *q;
     if (p->next == p) {
        free(p);
        p = NULL;
     } else {
        q = p->next;
        p->next = q->next;
        free(q);
      }
     return p;
}
```