Competitive Programming

(1) "\n" works faster than endl

(2) scanf and printf are usually slightly faster.

(3) int → $-2 \cdot 10^9$ ___ $2 \cdot 10^9$ (32-bit type)
mostly used ⇐ long long → $-9 \cdot 10^{18}$ ___ $9 \cdot 10^{18}$ (64-bit type)
__int128_t → -10^{38} ___ 10^{38} (128-bit type)

(4) Modular Arithmetic

$(a+b) \bmod m = (a \bmod m + b \bmod m) \bmod m$
$(a-b) \bmod m = (a \bmod m - b \bmod m) \bmod m$
$(a \cdot b) \bmod m = (a \bmod m \cdot b \bmod m) \bmod m$

~~Thus, we can take~~

(4) Floating Point Numbers:

double 64 bit
long double 80 bit → more accurate
less precession error.

→ A better way to compare floating pt numbers is to assume first two no are equal if the difference b/w them is less than $\approx 10^{-5}$.

Page No.
Date :

(5) Shortening Code

* typedef → used to give a short name to a data type.

eg.

```
typedef long long ll;
```

after this, code

```
long long a = 12345678;
long long b = 98765432;
cout << a*b << "\n";
```

can be shortened as follows.

```
ll a = 12345678;
ll b = 98765432;
cout << a*b << "\n";
```

→ typedef can also be used with more complex types. (eg. vectors, pairs)

eg.

```
typedef vector<int> vi;
typedef pair<int,int> pi;
```

typedef is limited to giving symbolic names to types only where as #define can be used to define alias for values as well. eg you can define 1.

Date :

* Macros → it specifies that certain strings in the code will be changed before the compilation.

```
#define PB push_back
#define MP make_pair
#define F first
#define S second.
```

After this, the code

```
V.push_back(make_pair(y1,x1));
V.push_back(make_pair(y2,x2));
int d = V[i].first + V[i].second;
```

can be shortened as follows:

```
V.PB(MP(y1,x1));
V.PB(MP(y2,x2));
int d = V[i].F + V[i].S;
```

→ A macro can also have parameters, which makes it possible to shorten loops and other structures.

eg- `#define REP(i,a,b) for(int i=a; i<=b; i++)`