MANAGING I/O OPEARTIONS

TOPICS

- INTRODUCTION
- CONSOLE I/O FUNCTIONS
- FORMATTED I/O FUNCTIONS
- UNFORMATTED I/O FUNCTIONS
- MCQS
- PROBLEMS

INTRODUCTION

- All input and output is performed with streams.
- A "stream" is a sequence of characters organized into lines.
- Each line consists of zero or more characters and ends with the "newline" character.
- ANSI C standards specify that the system must support lines that are at least 254 characters in length (including the newline character).
- Standard input stream is called "stdin" and is normally connected to the keyboard
- Standard output stream is called "stdout" and is normally connected to the display screen.
- Standard error stream is called "stderr" and is also normally connected to the screen.

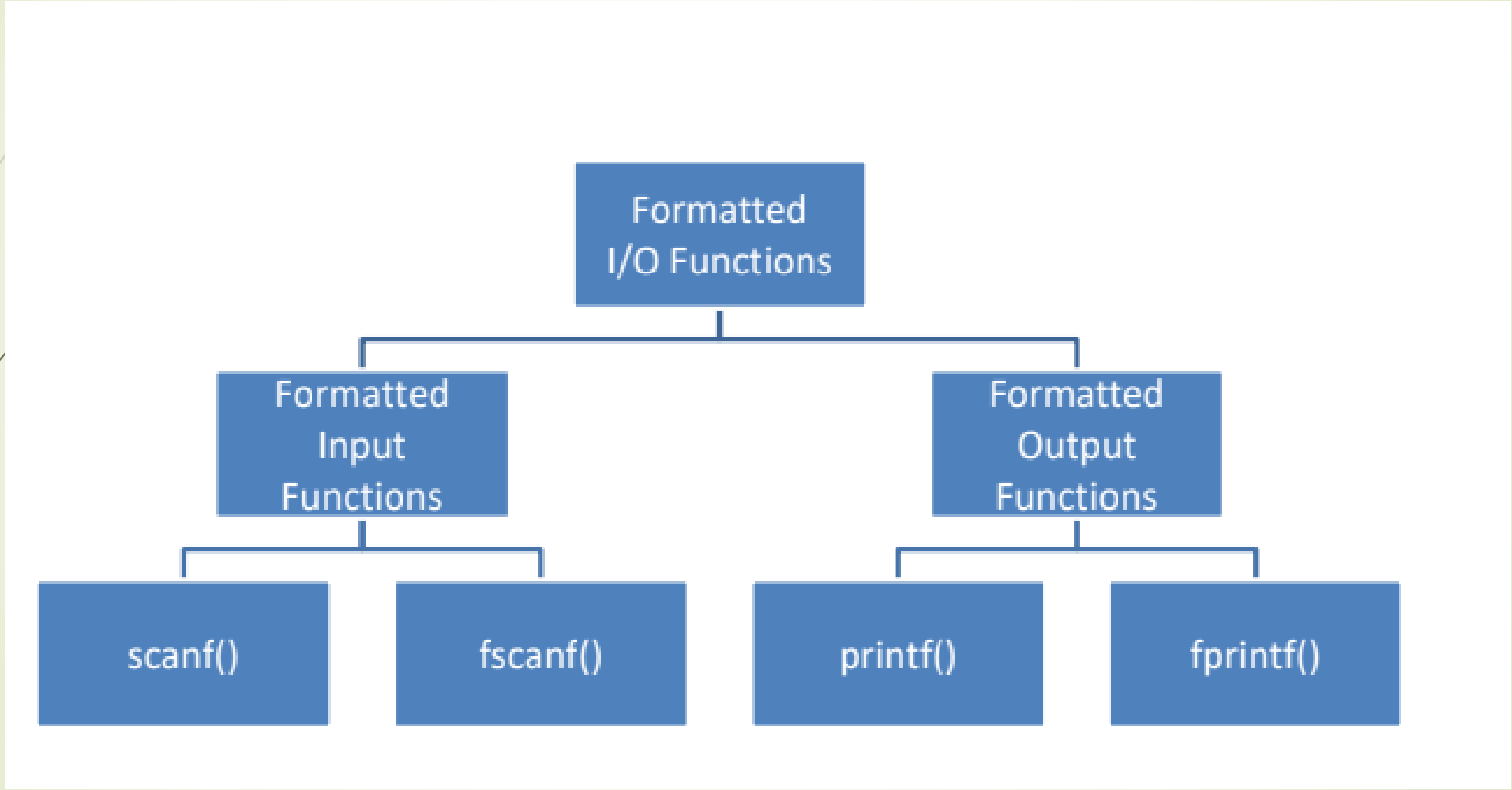
INTRODUCTION

- C Language has a set of library functions to perform I/O operations.
- The I/O library functions are listed the “header” file <stdio.h>.
- These libraries are available with all C compilers.
- There are numerous library functions available for I/O. These can be classified into two broad categories:
- (a) **Console I/O functions** - Functions to receive input from keyboard and write output to VDU.
- (b) **File I/O functions** - Functions to perform I/O operations on floppy disk or hard disk.

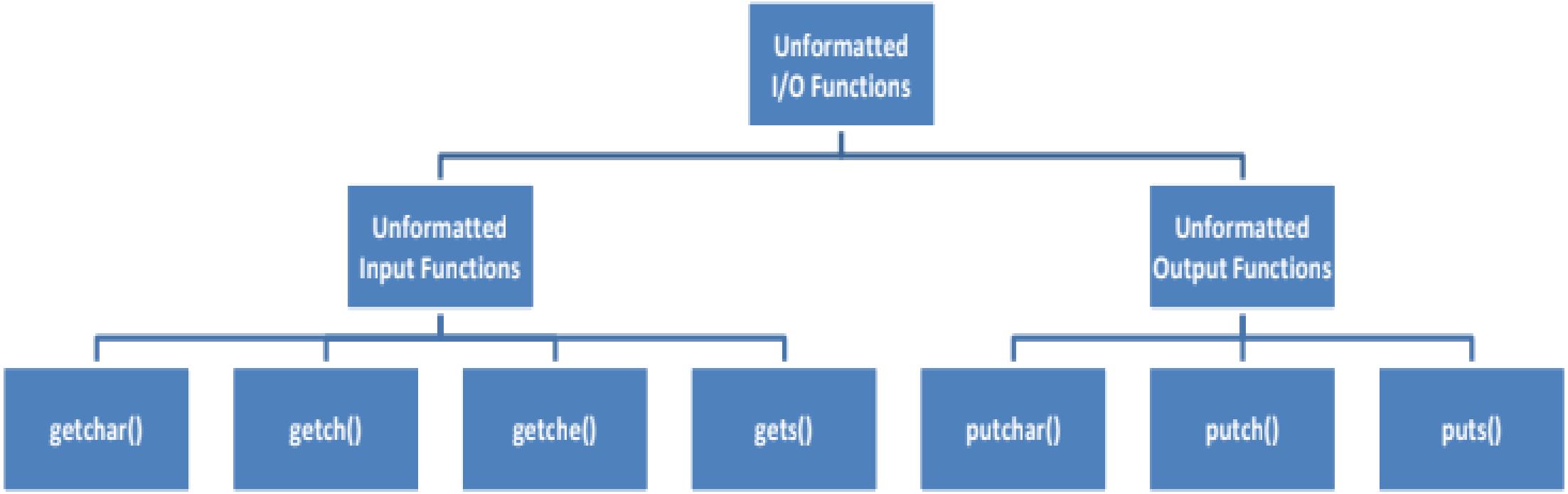
CONSOLE-I/O FUNCTIONS

- The screen and keyboard together are called a console.
- Console I/O functions is further classified into two categories,
  - **formatted console I/O functions.**
  - **unformatted console I/O functions.**
- **Formatted functions** allow to use the input got from keyboard and output displayed on VDU as per our requirements.
- For example, if values of average marks and percentage marks are to be displayed on the screen, then the details like where this output would appear on the screen, how many spaces would be present between the two values, the number of places after the decimal points, etc. can be controlled using formatted functions.
- In Unformatted functions ,no such things are taken into account.

CONSOLE-I/O FUNCTIONS



CONSOLE-I/O FUNCTIONS



FORMATTED CONSOLE-I/O FUNCTIONS- printf() function

- This function provides formatted output to the screen.
- Syntax:
- **int printf (“format”, var1, var2, …var n);**
- The “format” includes a listing of the data types of the variables to be output and, optionally, some text and control character(s).
- The return type of printf() is “int” and it returns ***total number of Characters Printed, Or negative value if an output error or an encoding error***
- Example:
- int b=10;
- printf (“The value of b is: %d \n”,b);// %d –format specifier of integer

FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-format specifiers

Objective-C Data Types Table

Type	Example	Specifier
char	'a', 'O', '\n'	%c
int	14, -14, 780, 0xEEC0, 098	%i, %d
unsigned int	10u, 121U, 0xEFu	%u, %x, %o
long int	18, -2100, 0xfeefL	%ld
unsigned long int	13UL, 101ul, 0xfefeUL	%lu, %lx, %lo
long long int	0xe5e5e5LL, 501ll	%lld
unsigned long long int	10ull, 0xffeeULL	%llu, %llx, %llo
float	12.30f, 3.2e-5f, 0x2.2p09	%f, %e, %g, %a
double	3.1415	%f, %e, %g, %a
long double	3.5e-5l	%Lf, %Le, %Lg, %La
id	Nil	%@

FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-optional specifiers

Specifier	Description
dd	Digits specifying field width
.	Decimal point separating field width from precision (precision stands for the number of places after the decimal point)
dd	Digits specifying precision
-	Minus sign for left justifying the output in the specified field width

FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-optional specifiers

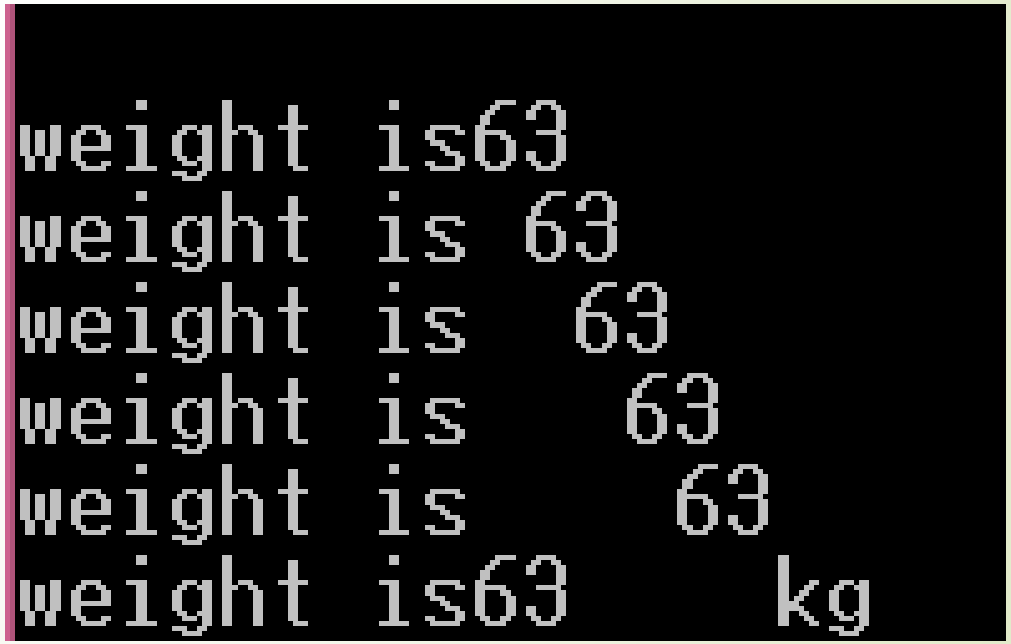
- The field-width specifier is used to decide how many columns should be used on the screen to print a value.
- Eg:%10d,prints the variable as a decimal integer in a field of 10 columns.
- If the value of variable is not up to fill the entire field then the value is right justified and padded with blanks on left.
  - Eg: int a=115,
  - printf("%10d",a);

- If the field specifier includes a minus the then value is left justified and padded with blanks on right.
- If the field width is less than what is required then it is completely ignored.

FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-Example

```
#include <stdio.h>
int main()
{
 int weight = 63 ;
 printf ( "\nweight is%2d", weight ) ;
 printf ( "\nweight is%3d", weight ) ;
 printf ( "\nweight is%4d", weight ) ;
 printf ( "\nweight is%5d", weight ) ;
 printf ( "\nweight is%6d", weight ) ;
 printf ( "\nweight is%-6dkg", weight ) ;
 return 0;
}
```

OUTPUT:



FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-Example

```
#include <stdio.h>
int main()
{
 printf ( "\n%10.2f%10.2f", 5.0, 13.5) ;
 printf ( "\n%10.2f%10.2f ", 305.0, 1200.9);
 return 0;
}
```

OUTPUT:

```
      5.00     13.50
    305.00   1200.90
C:\TDM-GCC-64\PL-Phase 1>
```

FORMATTED CONSOLE-I/O FUNCTIONS-printf() function-Example

```
#include <stdio.h>
int main()
{
 char ch = 'z' ;
 int i = 125 ;
 float a = 12.55 ;
 printf ( "\n%c %d %f", ch, ch, ch ) ;
 printf ( "\n%c %d %f",i ,i, i ) ;
 printf ( "\n%f %d\n", a, a ) ;
}
```

OUTPUT:

z 122 garbage value

} 125 garbage value

12.550000 garbage value

FORMATTED CONSOLE-I/O FUNCTIONS- sprintf() function

sprintf:

Syntax:

int sprintf(char *str, const char *string,...);

String print function instead of printing on console store it on char buffer which are specified in sprintf

```c
#include<stdio.h>
int main()
{
    char buffer[50];
    int a = 10, b = 20, c;
    c = a + b;
    sprintf(buffer, "Sum of %d and %d is %d", a, b, c);

    // The string "sum of 10 and 20 is 30" is stored
    // into buffer instead of printing on stdout
    printf("%s", buffer);

    return 0;
}
```

FORMATTED CONSOLE-I/O FUNCTIONS-
fprintf() function

fprintf is used to print the string content in file but not on stdout console.

int fprintf(FILE *fptr, const char *str, ...);

INPUT/OUTPUT FUNCTIONS

FORMATTED INPUT FUNCTION: scanf ()

This function provides for formatted input from the keyboard.

Syntax:

int scanf (“format” , &var1, &var2, …) ;

- The “format” is a listing of the data types of the variables to be input and the & in front of each variable name tells the system WHERE(address) to store the value that is input. It provides the address for the variable**.**
- **It returns total number of Inputs Scanned successfully, or EOF if input failure occurs before the first receiving argument was assigned.**

Example:

```
float a;  int b;
scanf (“%f%d”, &a, &b);
```

FORMATTED CONSOLE-I/O FUNCTIONS-sscanf() function

- The C library function **int sscanf(const char *str, const char *format, ...)** reads formatted input from a string.
- `int sscanf(const char *str, const char *format, ...)`

FORMATTED CONSOLE-I/O FUNCTIONS-
fscanf() function

- The C library function **int fscanf(FILE *stream, const char *format, ...)** reads formatted input from a stream.
- int fscanf(FILE *stream, const char *format, ...)

USAGE OF FORMAT SPECIFIER IN scanf() function

- **Example:**

```
#include<stdio.h>
 int main()
 {
int a;
scanf("input is : %d", &a);
printf("%d", a);
return 0;
 }
```

OUTPUT:

input is : 10

10

USAGE OF FORMAT SPECIFIER IN scanf() function

- **Example:**

// Extracting single digit using C Code.

```
#include<stdio.h>
  int main()
  { //1234
 int a,b,c,d;
 scanf("%1d%1d%1d%1d", &a, &b, &c, &d);
 printf("%d", a+b+c+d);
 return 0;
  }
```

OUTPUT:

1234

10

USAGE OF FORMAT SPECIFIER IN scanf() function

- **Example:**

// getting digit input in expression (a+b) format using C Code.

```
#include<stdio.h>
 int main()
 {
int a;
scanf("%d+%d", &a,&b);
printf("%d %d", a, b);
return 0;
 }
```

OUTPUT:

10+11

10 11

UNFORMATTED FUNCTIONS

getchar(), getch() and getche()

- All of these functions read a character from input and return an integer value.
- **getchar()** It reads a single character from the standard input and returns the corresponding integer value (typically ASCII value of read character) on success. It returns EOF on failure.

Syntax:

int getchar();

```
#include <stdio.h>
int main()
{
   printf("%c", getchar());
   return 0;
}
```

UNFORMATTED FUNCTIONS

getch(): getch() is a nonstandard function and is present in conio.h header file which is mostly used by MS-DOS compilers like Turbo C. It is not part of the C standard library or ISO C.

- Like above functions, it reads also a single character from keyboard. But it does not use any buffer, so the entered character is immediately returned without waiting for the enter key.

Syntax:

int getch();

```
#include <stdio.h>
int main()
{
   printf("%c", getch());
   return 0;
}
```

getche() is similar to getch();

UNFORMATTED FUNCTIONS

gets(): Reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached.

Syntax:

char * gets (char * str);

str :Pointer to a block of memory (array of char)

where the string read is copied as a C string.

returns : the function returns str.

```
int main()
{
    char buf[MAX];
    printf("Enter a string: ");
    gets(buf);
    printf("string is: %s\n", buf);

    return 0;
}
```

UNFORMATTED FUNCTIONS

The putchar(int char) :is used to write a character, of unsigned char type, to stdout. This character is passed as the parameter to this method.

Syntax:

int putchar(int ch);

This function returns the character written on the stdout as an unsigned char. It also returns EOF when some error occurs.

```
#include <stdio.h>
  int main()
{
    // Get the character to be written
    char ch = 'G';
     // Write the Character to stdout
    putchar(ch);
     return (0);
}
```

UNFORMATTED FUNCTIONS

puts() function is used to write a line to the output screen. In a C program, we use puts function as below. int

Syntax:

int puts(const char *string)

string – data that should be displayed on the output screen.

If successful, non-negative value is returned. On error, the function returns EOF.

```
#include <stdio.h>
#include <string.h>

int main()
{
   char string[40];
   scanf("%s",string);
   puts(string);
   return 0;
}
```

MCQS

What will be the output?

```
#include <stdio.h>
int main()
{
   printf("Hello ", printf("REC "));
   int i=printf("REC");
   printf(" %d",i);
   return 0;
}
```

.

OUTPUT:

REC Hello REC 3

MCQS

What will be the output?

```
#include <stdio.h>
int main()
{
   int j=printf("Hello", printf("REC "));
   int i=printf("REC");
   printf(" %d %d",j,i);
return 0;
}
```

OUTPUT:

REC Hello REC 5 3

MCQS

```
#include <stdio.h>
int main()
{
int i = 1, j = 4;
printf("%d %d %d", i, j);
return 0;
}
```

a) Compile time error

b) 1 4

c) 1 4 some garbage value

d) Undefined behaviour

OUTPUT:

Answer: c

MCQS

```
#include <stdio.h>
int main()
{
int i = 1, j = 4,k=1;
printf("%d %d ", i, j,k);
return 0;
}
```

a) Compile time error

b) 1 4

c) 1 4 some garbage value

d) Undefined behaviour

OUTPUT:

Answer: b

MCQS

What will be the output of the C program?

```
#include<stdio.h>
int main()
{
    printf("%d",printf("rec"));
    return 0;
}
```

A. compilation error

B. Runtime error

C. rec

D. rec3

OUTPUT:

Answer:D

MCQS

```
int main()
{
    int a = 4;
    printf("hello"+3);
    return 0;
}
```

A. compilation error

B. hellohellohello

C. hello

D. lo

E. None of the above

OUTPUT:

Answer:D

PROBLEM 1

- Roy wants to change his profile picture on Facebook. Now Facebook has some restriction over the dimension of picture that we can upload. Minimum dimension of the picture can be **L x L**, where **L** is the length of the side of square.
- Now Roy has **4** photos of various dimensions.
  Dimension of a photo is denoted as **W x H**
  where **W** - width of the photo and **H** - Height of the photo
- When any photo is uploaded following events may occur:
- [1] If any of the width or height is less than L, user is prompted to upload another one. Print "**UPLOAD ANOTHER**" in this case.
  [2] If width and height, are equal,
- (a) and equal to L,then it is accepted. Print "**ACCEPTED**" in this case.
  (b) else user is prompted to crop it. Print "**CROP IT**" in this case.
- (quotes are only for clarification)
- Given L, W and H as input, print appropriate text as output.

PROBLEM 2

- There are 7 floors in a flat and only 2 lifts. Initially Lift A is at the ground floor and Lift B at the top floor. Whenever someone calls the lift from N th floor, the lift closest to that floor comes to pick him up. If both the lifts are at equidistant from the N th floor, them the lift from the lower floor comes up.
- **INPUT**
- First line contains a single integer N denoting the floor from which lift is called.
- **OUTPUT**
- Output T lines containing one character "A" if the first lift goes to N th floor or "B" for the second lift.