Jai Ganesha

INDEX

NAME Vinay mallesh Hm STD ____ SEC ____ ROLL NO ____

S.No.	Date	Title	Page No.	Teacher's Sign/ Remarks

c:\user\vinay mallesh > g: — to select memory

q:\> CD C → to select folder

q:\c> gcc programname.c → program compile

to next folder selection we like

q:\c\basic > ~~cd~~ ~~scan~~ print start > cd..

then it will show

q:\c\basic > ~~cd scan start~~

↓ tree

q:\c\basic > CD scan start

q:\c > cd basic Print 1.

Printf() statement

used to display the value on output screen

Syntax: printf("%. format specifier", value);
↓ type of value
↓ output value

commonly used \n → new line \t → one tab space

%d → integer no
%f → float no
%c → char
%s → string

eg:- printf("%d\t", 12);
printf("%d", 34);

pre → #include
void main()
{
printf("employee id %d\n", 48376);
printf("employee name is %s\n", "Ramesh");
printf("employee salary is %f\n", 25,000.00);
printf("employee grade is %c\n", 'A');
printf("employee %s draws a salary of Rs %.2f\n", "Ramesh", 25'000.00);
// output: employee Ramesh draws a salary of 25000.00

correct

variable.

→ used to store the value

variable declaration

data type variable.
↓ identifier

type of value to be stored

int — eg:- int id;
float — float salary;
char — char grade;
string
long
:

the declaration is just done then initialization take place

variable initialization

variable name = value;

eg:- id = 4187;
salary = 25000.00;
grade = 'A';

int id = 4813;

eg:- int a1, a2, a3;
int a1, float a2, -> error
int a1, a1 = 10, a3;
int a1; float a2; ← termination.

```
if (exp1)
{
  statement 1;
  statement 2;
}
else if (exp 2)
{
  statement 3;
  statement 4;
}
else
{
  statement 5;
  statement 6;
}
```

working

is evaluate the exp 1.

if it result in true (1), excute the statement of body

if it result of expression 1 is false (0) evaluates expression 2

if result of exp 2 is true (1), excutes the statement of else if body

if ——— is ——— false (0), ——— else body

prog:-

```
#include <stdio.h>
void main()
{
  int num;
  printf("Enter a number\n");
  scanf("%d", &num);
  if (num > 7)
  {
    printf("number is above");
  }
  else
  {
    printf("number is below 7");
  }
}
```

correct

prg 2 :-

```
#include <stdio.h>
void main()
{
  int num;
  print("enter a number\n");
  scanf("%d", &num);
  if (num > 7)
  {
    print("number is above 7");
  }
  else if (num >= 7)
  {
    printf("number is equal to 7");
  }
```

correct

```
else
{
printf ("number is below 9");
}
}
```

```
#include <stdio.h>
void main()
{ float accbal=5000.00;
  float amt;
printf("welcome to cubcal banking system\n");
printf(" enter amount to withdraw\n");
scanf ("%f", &amt);
if (amt < accbal)
{ accbal = accbal - amt;
printf ("withdraw success\n");
}
else
printf ("withdraw failed\n");
printf (" insufficient balance, try later... !!\n");
}
printf("Account balance: %.f\n", accbal);
}
```

correct

```
#include<stdio.h>
void main()
{ float accbal = 5000.00
  float amt;
  int expin = 1234;
  int actpin;
printf(" welcome to cubcal banking system\n");
printf (" enter pin\n");
scanf (" %d", &actpin);
if ( actpin == expin)
{ printf (" enter amount to withdraw\n");
  scanf (" %f", &amt);
if ( amt < acc bal)
{
  accbal = accbal - amt;
  printf (" withdraw success\n");
} else{
  printf (" withdraw Failed:\n");
  printf (" insufficient balance, try later... !!\n");
```

group of if statement called nested if statement

correct

PRG :-> {

// variable declaration -> comment

int id;

float salary;

char grade;

// variable initialization

id = 4398;

Salary = 45553.40

grade = 'B';

printf("Employee id: %d\n", id);

printf("Employee salary: %.2f\n", salary);

printf("Employee grade: %c\n", grade);

}

correct

PRG -> {

int x = 10;

printf("x value is %d\n", x);

x = 20; // re-initialization.

printf("x value is %d\n", x);

x = 30; // re-initialization.

printf("x value is %d\n", x);

}

correct

PRG -> {

int x1 = 12;

int x2 = 34;

int x3 = 56;

printf("x1 value is %d\n", x1);

printf("x2 value is %d\n", x2);

printf("x3 value is %d\n", x3);

x2 = x1; // copy value of x1 to x2

x3 = x2; // copy value of x2 to x3

printf("x1 value is %d\n", x1);

printf("x2 value is %d\n", x2);

printf("x3 value is %d\n", x3);

}

correct

Reading i/p from keyboard

Bafna Gold Date: Page:

scanf("% format specifier", &variablename);
read user input & store in the variable

eg:- int id;
scanf("%d", &id); → C D .. ↓ next folder

```
pre:-{
    int id;
printf("enter Employee id\n");
scanf("%d", &id);
printf("id value is: %d \n", id);
}
```

correct — %d\n here we should give two time i/p value like eg: 123456 123456 then will display

```
pre {  //variable declaration → comment
    int id;
    float salary;
// read i/p from user
printf("enter the employee Id\n");
scanf("%d", &Id);
printf("enter the employee salary\n");
scanf("%f", &salary);
printf("Employee Id: %d \n", id);
printf("employee salary: %.2f \n", salary);
```

correct

operators - used to operate on values

1] Arithmetic operator → + - / * %
2] logic — " — → &&, ||, !
3] Relational — " — → >, <, >=, <=, !=, ==
4] Unary — " — → ++, --
5] Bit wise — " — → ~, >>, <<
6] Assignment — " — → =+, ---
7] Ternary — " — → expr1 ? expr2 : expr

```
pre:- {
    int  x= 10;
    int  y=5;
    int  r1, r2, r3, r4, r5;
```

```
r1 = x+y;
r2 = x-y;
r3 = x*y;
r4 = x/y;
r5 = x%y
printf("x=%d\n", x);
printf("y=%d\n", y);
printf("x+y=%d\n", r1);
printf("x-y=%d\n", r2);
printf("x*y=%d\n", r3);
printf("x/y=%d\n", r4);
printf("x mod y=%d\n", r5);
}
```

correct

Unary operator (one)

++ increment current value by 1

-- decrement current value by 1

eg:- 1) int x=0;
int y=0;
y = x++;
printf("%d\n", x);
printf("%d\n", y);

correct

assign
do first

Post increment
→ first use current value, then inc by 1

Pre - increment
→ first increment by 1, then the increment value

eg:- 2) int x=0;
int y=0;
y = x++ + x;
printf("%d\n", x);
printf("%d\n", y);

x = 01
y = 0 x++ + x
= 0 + 1 = 1

correct

3) int x=0;
int y=0;
y = x++ + x + x++ + x;
printf("%d\n", x);
printf("%d\n", y);

2
4

0+2+2+4+4+5+5   0+1+2=5 x=4

x=6

y=22

= 0+2+2+4 = 8

Ex:- int x=0;
int y=0; plus sign
y= x++ + ++x + x++ + ++x   Correct
printf("%d\n", x); 4
printf("%d\n", y); 8

int x=0   Correct
int y= 0
y = x++ + ++x + x++ + ++x + x++ + x + x++;
printf("%d\n", x); 6
printf("%d\n", y); 22

22 | 42

control statement

→ control the flow of execution

1) branching
(i) if stmt
(ii) switch stmt

2) looping
i) for loop
ii) while loop

if statement

Syntax 1:
```
if (Expression)
{   statement 1;
    statement 2;
}
```
if body

working
1) evaluates the Expression
2) if result in true (1), Executes the statement if body
3) if result in false (0), skips the if body statement

Syn 2:-
```
if (exp)
{   statement 1;
    statement 2;
}
else
{   statement 3;
    statement 4;
}
```

working
1) evaluates the expression
2) if result in true (1), Executes the statement if body
3) if result in false (0), Executes the else state. body statement.

```
}
printf("Account balance: %f\n", accbal);
} else {
printf("invalid pin\n");
}
```

Switch statement — 25/02/19

Syntax:-
```
switch (expression)
{
case value1: stmt 1;
             stmt 2;
             break;
case value2: stmt 3;
             stmt 4;
             break;
case value3: stmt5;
             stmt6;
             break;
default: stmt 7;
         stmt 8;
}
```

working
1) evaluate exp
2) searches the matching case, executes the matched case body statement
3) no case one matched, executes the default case.

eg1:-
```
{
char grade;
printf("enter a grade\n");
scanf("%c", &grade);
switch (grade)
{
case 'A': printf("First class with distinction\n");
          break;
case 'B': printf("First class\n");
          break;
case 'C': printf("second class\n");
          break;
case 'D': printf("just pass\n");
          break;
case 'E': printf("not last\n");
default : break
          printf("invalid grade\n");
}
```

(correct)

Prog 2:-

```
#include <stdio.h> void main()
{
float acc bal = 10000.00;
int expin = 1234;
int actpin;
```

correct

```
printf("welcome to online banking\n");
printf("enter the pin\n");
scanf("%d", &actpin);
if (actpin == expin);
{
int input;
float amt;
printf("type 1 to view balance\n");
printf("type 2 to deposite\n");
printf("type 3 to withdraw\n");
scanf("%d", &input);
switch (input)
{
case 1: printf("account balance: %.2f\n", accbal);
        break;
case 2: printf("enter amount");
        scanf("%f", &amt);
        accbal = accbal + amt;
        printf("Account balance: %.2f$\n", accbal);
        break;
case 3: printf("enter Amt");
        scanf("%f", &amt);
        if (amt < accbal)
        {
        printf("withdraw success\n");
        acc bal = accbal - amt;
        }
        else
        {
        printf("withdraw failed\n");
        printf("insufficient balance\n");
        }
        break;
}
```

Bafna Gold
Date: Page:

02/19

Exp
the matchi
executes the
body

e matched,
default

tion\n");

```
}
else
{
printf("invalid pin\n");
}
}
```

Loop

for loop statement

```
for( initialization ; condition; inc/dec)
{
    statement1 ;
    stmt 2 ;
}
```

① ② ③ ④

Execution sequence: ① → ② → ④ → ③ → ② (true) … (false)

iteration

```
#include <stdio.h>
void main()
{
int i;
```

Prog 3:→ Eg:-

```
for (i=1 ; i<=5 ; i++)
{
printf(" I love programming ");
}
```

correct

```
#include <stdio.h>
void main()
{
int i;
```

Prog 4:-

```
for(int i=1; i<=10 ; i++)
{
int res= i*i;
printf("square of %d is %d\n", i, res);
}
}
```

correct

o/p

```
I Love Prem
I Love Prem
I Love Prem
I Love Prem
I Love Prem
```

o/p

```
1 2 3 4 5 6 7 8 9 10
```

Prog 5:- write a prem display only even numbers 1 to 20

```
{ int i;
for (i=1 ; i<=20 ; i++)
{
int res = if (i%2==0)
print ("%d\n", i);
}}
```

correct

write A program to withdraw Rupees 2000 from a bank account & find & after call the withdraw display account balance.

pseudo account balance can be assumed any value.

```
{
float accbal=15000.00;
float amt = 2000.00;
for (int i=1; i<=5; i++)
{
accbal = accbal - amt;
```

eg:-

```
float accbal=5000.00;
float amt=2000.00;
for (int i=1; i<=5; i++)
{
if(amt <= accbal)
{
printf("withdrawing Rs %.2f\n", amt);
accbal = accbal-amt;
}
else
{
printf("insufficient balance\n");
break;
}
}
printf("Account Balance: %.2f\n", accbal);
```

Correct

~~Array in C~~ 26/02/19

while loop

working

syntax :- while (condition)
{
stmt 1;
stmt 2; } iteration
stmt 3;
}

1] evaluate condition
2] if true, execute the body of...
3] Repeat the iteration until condition become false.

e.g :- int i = 1;
while (i <= 5)
{ printf ("%d\n", i);
i++; }

correct

prog :- printf ("main function started\n");
// declare array
int size;
printf ("enter the array size\n");
scanf ("%d", &size);
int a[size]; // array of int type
// code to read value and store in array
int i = 0;
while (i < size)
{
int temp;
printf ("Enter value\n");
scanf ("%d", &temp);
a[i] = temp;
i++;
}
printf ("displaying array element\n");
int j = 0;
while (j < size)
{
printf ("%d index value %d\n", j, a[j]);
j++;
}
printf ("main function end\n");
}

correct

① int i=1; → [correct] if we left blank it will automatically take zero
for(; i<=5; i++)
{
printf("%d\n", i);
}

② int i=1;
for(; i<=5;) → [correct]
{
printf("%d\n", i);
i++;
}

③ int i=1; → true [correct]
for(; ;)
{
printf("%d\n", i);
i++;
if(i>5)
{
break;
}
}

④ int i=1; → by default true
while()
{printf("%d\n", i);
i++;
if(i>5)
{
break;
}
}

→ write a prm to display all the even values present in a array. create your own array & values.

```
int size;
printf("Enter array size\n");
scanf("%d", &size);
int a[size];
{
for(int i=0; i<size; i++);
{
if(a[i]%2 ==0){
printf("%d\n", a[i]);
}
}
}
```

→ write a prog to display even position of an value an array.

```
for (int i=0 ; i< size ; i++)
{
  if ( i%2 == 0 )
  { printf("%d\n", a[i]);
  }
}
```

→ write a prog to display those element where index also even and value also even.

```
int a[size]
for ( int i=0;  i< size ;  i++)
{
  if (a[i]%2==0 && i%2==0)
  {
    printf("%d\n", a[i]);
  }
}
```

→ Repeated no finds a[i] delete

```
int num, count = 0;
scanf("%d", &num);
for(int i=0; i<size; i++)
{
  if (num == a[i])
  {  count ++;
  }
}
printf("%d, occurs %d times.", num, count);
```

function

Bafna Gold — Date: 28/2/19

-> block of reusable code developed to perform the task.

-> achive reusability

-> achive modularity
  (break bigger task to smaller task)

-> a function should perform only one task/operation
eg: printf();
scanf();

Types of function

1) builtin function :- pre-difined by the language
eg :- printf(), scan()

2) user define function :- defined by the user/programmer

Libraries : -> collection of reusable functions.

① general purpose library :-
* provide by the language
* can be used in the any project

② project specific library
* developed by the developer/programmer
* can be used in only that project

function defenation syntax

(a) function declaration
(b) function definition

if u want declare

<access specifiers> <access modifier> return type functionName(argument list)
{
______
______ } to do operation
______
function body or defenation
return value;
}

prog1:-
```
// user defined function
void test()
{
printf("running test() function\n");
}
// main function - Entry point for execution
void main()
{
printf("main function started\n");
test(); // function call
test(); // function call
printf("main function ended\n");
}
```

access calling

for other function we should call that name

* whenever begin the an execution the execution unit is make a call to the main function of the program.
* the main function is a entry point for execution purpose. if we don't define a main function we can't begin the execution.
* the user defined function can executed only when the function is called for execution. an function can be called by using function name
* the function can be called from another function called.
* the function can be called any no of time's.

prog2:-
```
#include <stdio.h>
void disp()
{
printf("running disp() function\n");
}
// user defined function
void test()
{
printf("test() function started\n");
disp();
printf("test function ended\n");
}
```

prog3:-

prog4:-

// main function - Entry point for execution
void main()
{
printf("main function started \n");
test();
printf("main function ended \n");
}

prog3:- #include <stdio.h>
void square()
{
int num = 5 // hard coded prog
int res = num * num;
printf("square of %d is %d \n", num, res);
}
void main()
{
printf("main function started \n");
square();
square();
printf("main function ended \n");
}

prog4:- #include <stdio.h>
void square (int num)
{ int res = num * num;
printf("square of %d is %d \n", num, res);
}
void main()
{
printf("main function started \n");
square(5);
square(8);
square(10);
printf("main function ended \n");
}

op 25
64
100

Prog4:-

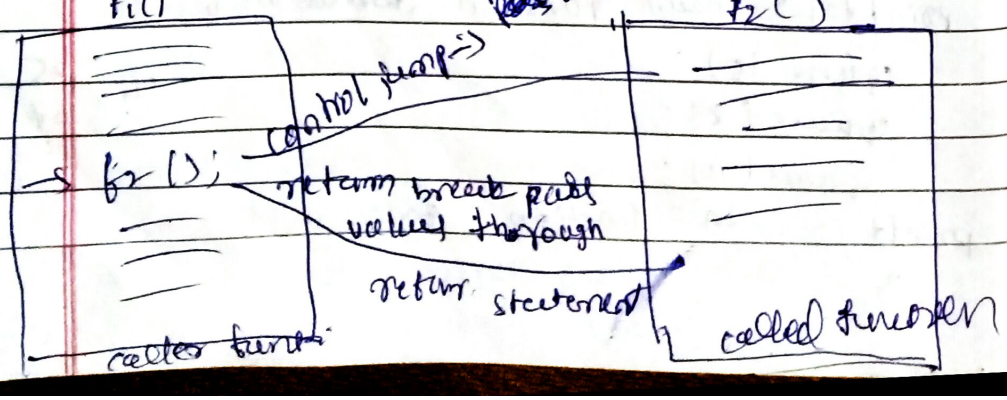
```
void square (int num)        → must be same
{ int res = num * num;          name
printf ("square of %d is %d \n", num, res);
}
void main ()
{
printf ("main function started \n");
int x;
printf ("enter a number \n");
scanf ("%d", &x);
square (x);
printf ("main function ended \n");
}
```

Prog5:-

```
#include <stdio.h>
void sum (int num1, int num2)      this int num1, num2
{                                  Because this is arguments
  int res = num1 + num2;
printf ("sum of %d and %d is %d \n", num1, num2, res);
}
void main()
{
  int x,y;
printf ("enter a number \n");
  scanf ("%d", &x);
printf ("enter a number \n");
  scanf ("%d", &y);
    sum (x,y)
}
```

(you should declare seperately not like)

function creation




```
#include <stdi
int test()
{ printf ("t
  return
}
void ma
{
printf(
int
printf
printf
}
```

Prog 2:- #inclo

```
int sq
{
(int
  retu
or -> // s
}
void
{
  int
printf
scan
int
print
}
funct
can
```

#include <stdio.h> → prog1

int test()
{ printf("test() running\n");
return 42
}

void main()
{
printf("main function started\n");
int var1 = test(); // assign return value of function
printf("var1 value is %d\n", var1);
printf("main function ended\n");
}

storing the value in var1 and the value is int data type so we declared.

Bafna
Date: 1/03/19

prog2:- #include <stdio.h>
int square (int num1) return value
{
int res = num * num;
return res;
or → // return num*num;
}
void main()
{
int x;
printf("enter a number\n");
scanf("%d", &x);
int res = square(x);
printf("square of %d is %d\n", x, res);
}

int res

function can return only one value, but a function can have any no of argument

Local variable

* if a variable declared in function body it is known as "local variables". scope of the variable is limited to the function body. the local variables can't be access in other function body
* A variable declared no outside the function known as "global variables"
* the global variable can be access by any function defined in the programmer.

prog :-

```
#include <stdio.h>          // local variable example
void f1()
{
int x = 12; // local variable
printf("x value : %d\n", x);
}
void f2()
int y = 34; // local variable
printf("y value : %d\n", y);
}
void main()
{
  printf("main function is started\n");
  f1();
  f2();
  printf("main function ended\n");
}
```

global variable example

```
#include <stdio.h>
const float pi = 3.14;
float area_circle (float real)
{
  float a1 = 2*real;
  return a1;
}
```

```
float area (float rad)
{
 float a1 = pi * rad * rad;
 return a1;
}
float circumference (float rad)
{
 float a1 = 2 * pi * rad;
 return a1;
}
void main ()
{
float rad, r1, r2, r3;
printf (" enter radical \n");
scanf ("%f", &rad);
r1 = diameter (rad);
r2 = area (rad);
r3 = circumference (rad);

printf (" Radical : %.2f \n", rad);
printf ("Diameter : %.2f \n", r1);
printf (" area : %.2f \n", r2);
printf (" circumference : %.2f \n", r3);
}
```

because return type

float ಬಳಸಿದಾಗ ಹಿಂದೆ return type ಕೊಡಬೇಕು
void ಇದ್ದಾಗ ಕೊಡಬೇಕಾಗಿಲ್ಲ function ಹೆಸರಿನ ಮುಂದೆ circumference function ನ float ಬರುವ return ಮಾಡಿ ಕೊಟ್ಟಿದ್ದೇವೆ.

Prog:- recursive

```
#include <stdio.h>
void test ()
{
 printf (" running test () function \n");
 test (); // calling itself
}
 // recursive call
void main ()
{
 printf (" main function started \n");
 test ();
 printf (" main function ended \n");
}
```

Prg:-
```
#include <stdio.h>
void test(int num)
{
    printf("%d", num);
    num++;
    if(num<=10)   // for recursive   if(num)
    test(num); // call itself
}
// recursive call
void main()
{
    printf("main function started\n");
    test();
    printf("main function ended\n");
}
```

Array

35 des salgrade

Name	Null?	Type
GRADE		number
LOSAL		—
HISAL		—

35 desc dual;

Name	Null?	Type
~~DUMMY~~		VARCHAR2(1)

array inC

```
#include<stdio.h>
int main()
{
printf("main function started\n");//declare array
    int a1[5]; // array of int type, can hold 5 int values
// storing values in array
a1[0] = 12;          or int a1[] = {12, 34, 53, 78, 82};
a1[1] = 34;          to reduce steps
a1[2] = 53;
a1[3] = 78;
a1[4] = 82;
printf("3rd element value is %d\n", a1[3]);
 printf("array elements\n");
  for(int i=0; i<5; i++){
    printf("%d index value is %d\n", i, a1[i]);
}
printf("main function end\n");
}
```

```
#include<stdio.h>
int main()
{
printf("main function started\n");
    //declaration array
int size;
printf("enter array size\n");
scanf("%d", &size);
 int a1[size]; //array of int type
 // code to accept values and store in array
```

```
for(int i=0 ; i< size ; i++)
  {
    int temp;
  printf("enter values\n");
   scanf("%d", &temp);
   a[i]= temp;
  }
print("Displaying array elements\n");
for(int j=0; j<size ; j++);
  {
    printf("%d index value is %d\n", j, a[j]);
  }
    print("main function end\n");
}
```

24 | 2 | 18

pointer

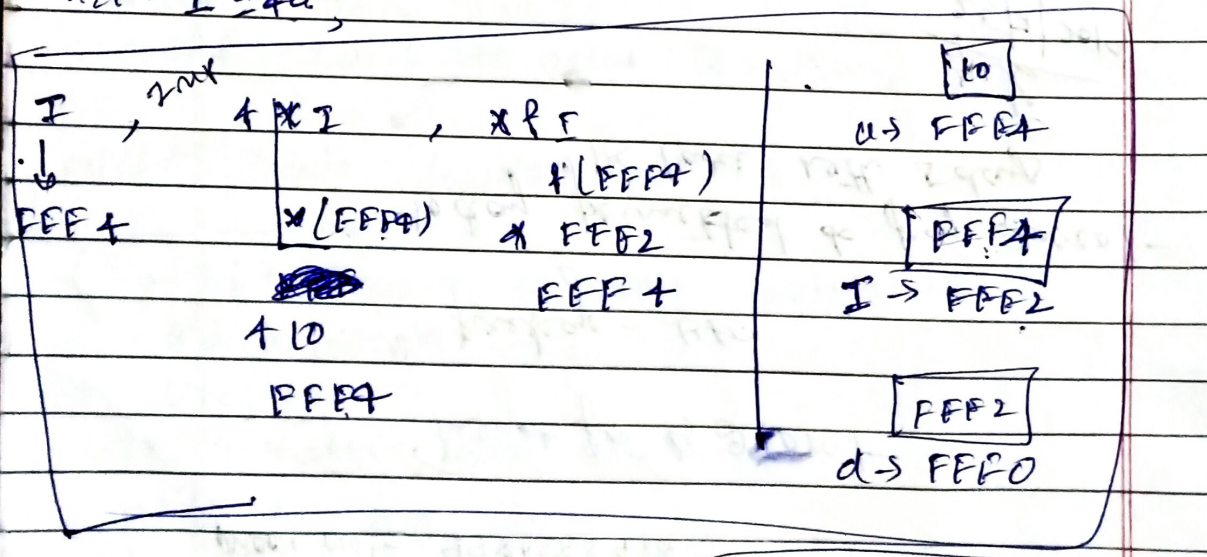
pointer is used to access the address.

```
int main()
{
 int a=10
  a//10
  &a//1000
int *I =&a;
```

10
a→1000

1000
I=2000

I // 2000
&I // 2000
*I // 10
*1000



& → address
* → address stored content

10

a → FFF4

FFF4

I → FFF2

FFF1

a → FFF0

10.

a → FFF4

FFF4

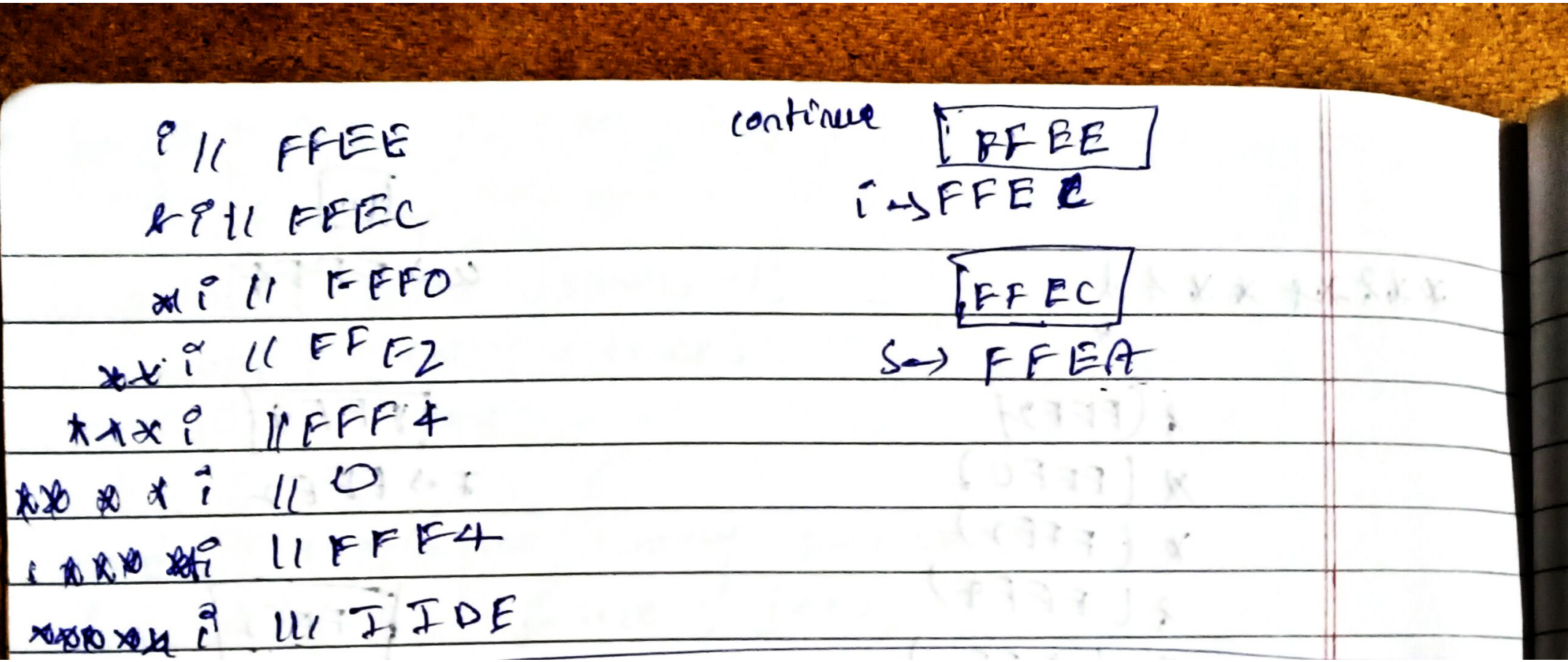
I → FFF2

FFF2

a → FFF0

FFF0

Y → FFEE


FFEE
FFEC
FFF0
FFF2
FFF4
0
FFF4
IIDE
continue
FFEE
FFEC
FFEA