1. Answer the following:
    a. i
    b. iv
    c. pointer code:
        i. 123
        ii. 10
        iii. 10
        iv. 123
        v. 10
        vi. Garbage value
        vii. 789
        viii. 123
2. Output and explanation:
    a. **ABC**: since i is initialized to 100, it is not equal to 10. In the second line of the code, a conditional/ternary operator is used. It checks the condition and finds it to be true. So it prints ABC. If it was false, DEF would be printed.
    b. **Garbage value 200**: In the first line of code, variable a is initialized to 100, b and c are declared but not initialized. The second line checks if a is greater than or equal to 500, using an if-statement. Since this condition is not satisfied, 300 is not assigned to variable b. In the absence of second bracket, only the next one line of code is considered part of if-statement. So, 200 is assigned to variable c. So, b contains garbage value and c contains 200 which are printed.
    c. Nothing is printed: variable i is initialized to 100. In the while loop, condition is checked and loop is entered only if i is less than or equal to 10. Since this condition is not met, while loop is not entered and nothing is printed.
    d. 14 13 12 11 10 :

Iteration	i value	i>=1 Condition satisfied?	Updated i value	Print
1	5	yes	4	14
2	4	yes	3	13
3	3	yes	2	12
4	2	yes	1	11
5	1	yes	0	10
6	0	NO → stop – do not enter loop		

    e. 0 5: integer variable a is initialized to 0. In the first if statement, the value of a is checked whether equal to 0, using relational operator ==. Since this condition is satisfied, only the next line, which is a part of this if statement in the absence of {} is executed. So 0 is printed. Next, another if statement is used, where the logical NOT operator is used (!) to evaluate !a as 1, since a is 0. This is because, if the condition is true then the logical NOT operator will make it false and vice-versa. So, this if is evaluated as if(1), which is always TRUE. So the next printf statement is executed. Since a=0, a+5 is evaluated to 5. So 5 is printed.

3. The code:

```c
#include <stdio.h>
void func(){
    float a,b;
    printf("Enter two numbers for multiplication: ");
    scanf("%f %f", &a, &b);
    printf("\n The product is: %f",a*b);
}
int main() {
    func();
    return 0;
}
```

4. The code: The values of m,p,and c variables need to use input. I have taken constant values here due to non-availability of space.

```c
#include <stdio.h>
int main(){
    int m=70, p=70, c=70, tot1, tot2; //m=10 and 70
    tot1 = m+p+c;
    tot2=m+p;
    if((m>=60&&p>=50&&c>=40&&tot1>=200)||tot2>=150)
        printf("Admit to college");
    else
        printf("Do not admit to college");
    return 0;
}
```

5. The code:

```c
#include <stdio.h>
void sum_n(int n, int *s){ //function with return type void
    int i, m=n;
    for(i=1;i<=n;i++)
        *s = *s+i; //calculating sum of n natural numbers
}
int main() {
    int n, sum=0;
    printf("Enter the value of n: "); //take user input from main
    scanf("%d",&n);
    sum_n(n, &sum); //mixed call - function call
    printf("\nThe sum of first n natural numbers = %d ", sum);
    //printing result in main()
    return 0;
}
```