Ans. to the Q no. 1

a) The difference between "H" and 'H' is that "H" represents a string • and this contains "NULL" '\0' at the end. On the other ~~bad~~ hand 'H' is considered to be a character which is the main difference between "H" and 'H'.

b)

```
char s1[30] = "Den";
char s2[30] = "Marking";
strncat (s1, s2, 4);
printf ("%s", s1);
```

The output will be DenMark;

c)

```
#include <stdio.h>
#include <string.h>
int Palindrome (char word[])
{
  char reverse[100];
  strcpy (reverse, word);
  strrev (reverse);
 if (strcmp (word, reverse) == 0)
 {
  return 1;
 }
```

```
    else
      {
       return 0;
      }
   }
int main () {
    char word [100];
    scanf ("%[^\n]", word);
    if ( Palindrome (word))
     {
      printf ("%s is a palindrome.", word);
     }
    else
      {
      printf ("%s is not a palindrome.", word);
      }
 return 0;
}
```

Ans. to the Q no. 1 (Or)

The difference in a string and a character is every string is a character string but not every character is string. Strings must be null-terminated while character arrays may or may not be null-terminated. Strings often involve using standard

manipulations function, while characters arrays are generally manipulated using apray operations.

b)

```
char s1[30] = "Bad";
char s2[30] = "Good";
strncpy(s1, s2, 2);
printf("%s, s1);
```

The output will be "God";

c)

```
#include <stdio.h>
int equalstrings(char str1[], char str2[]){
  int i = 0;
  while (str[i] != '\0' && str2[i] != '\0'){
    if (str1[i] != str2[i]){
      return 0;
    }
    i++;
  }
  if (str1[i] == '\0' && str2[i] == '\0'){
    return 1;
  }
  else {
    return 0;
  }
}
```

Ans. to the Q no. 2

```
#include <stdio.h>
struct Employee {
 int employeeID;
 double salary;
 int age;
};
 int main () {
 int N;
 scanf ("%d", &N);
 struct Employee employees [N];
 for (int = 0; i<N; i++)
  {
   scanf ("%d", &employees[i].employeeID);
   scanf ("%lf", &employees [i]. employee salary);
   scanf ("%d", &employees [i]. age);
  }
 double maxSalary = 0;
 for (int i = 0; i< N; i++)
  { if (employees [i]. age >= 40 && employees [i]. salary
                                    > max salary)
    {
     maxSalary = employees [i]. salary;
    }
  }
 printf ("Maximum salary age : %.2lf \n", maxsalary);
 printf ("size of the structure %lu bytes\n", sizeof(
                                    struct Employee));
 return 0;
}
```

Ans. to the Q no. 2 (On)

```
#include <stdio.h>
#include <string.h>
struct star {
    int starID;
    char name[50];
    int age;
};
int main() {
    int N;
    scanf("%d", &N);
    struct star stars[N];
    for (int i=0; i<N; i++)
    {
        scanf("%d", &stars[i].starID);
        scanf("%s", stars[i].name);
        scanf("%d", stars[i].age);
    }
    int youngest = stars[0].age;
    for (int i=1; i<N; i++)
    {
        if (stars[i].age < youngest)
        {
            youngest = stars[i].age;
        }
    }
    printf("Age of the youngest star %d\n", youngest);
    printf("size of the structure %lu bytes\n", sizeof(struct
                                                    star));
    return 0;
}
```

Ans. to the Q no. 3

a)
The value of

b = ++a[i]; ⇒ 8;

c = a[i++]; ⇒ 8;

d = a[i]; ⇒ 8;

b)
```
#include <stdio.h>
int n;
scanf("%d", &n);
int arr[n];
for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}
int oddCount = 0;
for (int i = 0; i < n; i++) {
    if (arr[i] % 2 != 0) {
        oddCount++;
        printf("%d", arr[i]);
    }
}
printf("\n Number of odd number : %d\n", oddCount);
return 0;
}
```

c)
```
#include <stdio.h>
int main () {
    char name [100];
    scanf ("%s", name);
    int n;
    scanf ("%d", &n);
    char filename [100];
    sprintf (filename, "%s.text", name);
    FILE* file = fopen (filename, "w");
    for (int i = 2; i <= n; i += 2)
    {
        fprintf (file, "%d\n", i);
    }
    fclose (file);
    return 0;
}
```

Ans. to the Q no. 4 (a)

A function prototype is a declaration of a function that specifies it's name, return type, and the types of its parameters. It provides essential information to the compiler about the functions interface without detailing its actual implementation. Function prototypes are usually placed at the beginning of a program or in header files.

b) Header files in C serve as a way to declare the structure and prototype of functions before they are used in a program. They provide essential information to the compiler about various function, data types and macros used in the program, helping ensure proper compilation and linking of code.

c)

```
#include <stdio.h>
int Maximum (int a, int b, int c)
{
    int max = a;
  if (b > max) {
      max = b;
    }
    if (c > max) {
      max = c;
    }
return max;
}
int main () {
    int result = Maximum (int a b, c);
    printf ("Maximum: %d\n", result);
    return 0;
}
```

Ans. to the Q no. 5

a) We cannot omit both row (3) and column (4) while declaring array because the compiler needs this info. to allocate the correct amount of memory for the array. The sizes of all dimensions are necessary for proper storage allocation and addressing of elements within array.

b)

4	1	0	2
-1	2	4	
0	-1		

since the size of rows isn't initialized properly & the input leaves those spaces empty, their values would be whatever happened to be in those memory locations at the time of creation. These values are unpredictable and may contain garbage values.

c)
```
#include <stdio.h>
int main () {
int N, M, X;
scanf ("%d %d", &N, &M);
int arr [N][M];
for (int i=0; i<N; i++)
{
 for (int j=0; j<M; j++)
  {
   scanf ("%d", &arr [i][j]);
  }
}
 scanf ("%d", &X);
int found = 0;
for (int i = 0; i < N; i++)
  {
  for (int j=0; j<M; j++)
    {
     if (arr [i][j] == X)
       {
        printf ("%d found at row %d column %d \n",
                                          X, i, j);
         found = 1;
         be break;
       }
    }
   if (found == 0) {
     break;
   }
}
return 0; }
```

Ans. to the Q no. 6

a)

If the address of x is 500,

i) p+3 => 500+(3*4) => 512;

ii) *p+1 => 16;

iii) *(p+1) => 6;

iv) *(p+1) -*p => -9;

v) --(*p) => 14;

b) The data type in-front helps the compiler understand the size and type of data that the pointer is expected to handle, which is crucial for correct pointer arithmetic and derefrencing operations. Here char *c helps the *c pointer help understand that it has to deal with character variables and same goes with int *ip.

c) if the address of a is 300 and b is 400,

	a	b	a_p	b_p
int a = 5, b = 10;	5	10	-	-
int * a_p = &a, * b_p = &b;	5	10	300	400
a = b + * a_p;	15	10	300	400
a_p = b_p	15	10	400	400
b = (* a_p) * (* b_p);	15	150	400	400
* b_p = a / b;	15	0	400	400
* a_p = a % b;	0	0	400	400

Ans. to the Q no. 1

a)
```
#include <stdio.h>
int main (){
  int i=0;
  char sen [1000]; char L [1000] = sen [0];
  scanf (" %[^\n]", sen);
  while (sen[i] != '\0')
  {
   if (sen[i]== 'a' || sen [i]== 'e' || sen [i] == 'i' || sen [i]=='o'
       || sen [i] == 'u' || sen [i] == 'A' || sen [i] == 'E' || sen[i]
       == 'I' || sen [i] = 'O' || sen [i] = 'U')
     {
       printf ("%s", sen [i]);
       i++;
     }
   else
      {
       i++;
      }
   i = 1;
  while (sen [i] != '\0')
   {
     if (sen[i]1]
     if (sen [i] == 'l' || sen[i] == 'L')
     {
        L[i] = sen [i];
        i++;
     }
   else
     {
      i++;
     }
```

```
    }
    printf("Last of occurrence of L is at position
            %d", i);
    return 0;
}
```

b) Output of every updated string will be,

```
char s1[30] = "johny";
char s2[30] = "pa";
char s3[30] = "yes";
char s4[30], char s5[30];
```

1. strcat (s1, " "); => "johny "
2. strcpy (s4, s1); => s4[30] = "johny ".
3. ~~strcpy~~ strcat (s4, s1); => "johny johny ".
4. strcat (s4, s3); => "johny johny yes ".
5. strcpy (s5, ""); => s5[30] = "";
6. strncat (s5, s2, 2); => "pa";
7. strcat (s5, s2); => "papa";
8. Strcat (s4, s5); => "johny johny yes papa".
9. printf ("%s", s4); => "johny johny yes papa.

Ans. to the Q no. 1 (or)

a)

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char Toge [1000];
  int len, i=0,
  scanf (" %[^\n]", Toge);
  len = strlen (Toge);
  while (Toge [i] != '\0')
  {
    if (len <= 10
  if (len<=10)
  {  while (Toge[i]! = '\0')
    {
      if (Toge[i]>='a' && Toge[i]<='z')
        {
          printf ("Curse got activated")
          i++;
        }
      else
        {
          printf ("Curse backfired \n");
        }
    }
  printf ("Curse got activated");
}
```

b) The "strcmp" library function is used to compare to strings and returns an integer value like the result. If the result is 0, it means the string are equal and if it's higher or lower than 0, then the string are not equal. The difference between "strcmp" and "strcmpi" is that "strcmp" is case sensitive & "strcmpi" is not case sensitive.

Ans. to the Q no. 2

a)

```
#include <stdio.h>
  int isLeapyear (int n)
   {
     if ((n%.4 == 0) && (n %. 100 != 0) || (n %. 400== 0)) {
        return 1;
      }
     else
      {
       return 0;
      }
}

int main () {
  int a, b ;
  scanf ("%.d %.d, &a, &b) ;
  for (int i = a ; i <= b ; i++)
  {  if (is Leap year (i))
     {
```

```
            printf ("%d\n", i);
          }
        }
    return 0;
}
```

b)

```
#include <stdio.h>
#include <string.h>
int main ()
{
char inp[1000]; char rev[1000];
 scanf (" %.[^\n]", inp]
  strcpy (rev, inp);
  strrev (rev);
 if (strcmp (inp, rev) == 0)
  {
   printf ("%.s is a palindrome\n", inp);
  }
 else
   {
    printf ("%.s is not a palindrome \n", inp);
   }
 return 0;
}
```

Ans. to the Q no.3

a)

if address of x is 700,

I) p+1 = 704,

II) *p-20 = -12,

III) *(p+1) = 7,

IV) *(p-1)-*p = 9,

V) ++(*p) = 9.

b)

```
# include <stdio.h>
  int main()
  {
    int mat[10][10];
    int n, i, j, sum=0;
    scanf("%d", &n);
    for(i=0; i<n ; i++)
      {
       for(j=0 ;j<n ; j++) {
           scanf("%d", mat[i][j]);
          }
       }
      for(i=0 ; i<n ; i++)
       {
          sum += matrix[i][i];
       }
```

```
    printf("Sum of the diagonal elements %d", sum);
    return 0;
}
```

Ans. to the Q no. 4

a)

```
#include <stdio.h>
int main ()
{
  int n, grade, countA = 0; countB = 0; countC = 0;
  int countD = 0, countF = 0;
  scanf ("%d", &n);
  for (int i = 0; i<n; i++)
  {
    scanf ("%d", & grade);
    if (grade >= 90 && grade <= 100)
    {
      countA++;
    }
    else if (grade >= 80 && grade <90)
    {
      count B++;
    }
    else if (grade >= 70 && grade < 80)
    {
      countc++;
    }
```

```
    else if (grade >= 60 && grade < 70)
    {
      count D++;
    }
    else if (grade >=0 && grade < 60)
    {
      count F++;
    }
  }
  printf ("The grade stats : \n");
  printf ("A : %d \n", count A);
  printf ("B : %d \n", count B);
  printf ("C : %d \n", count C);
  printf ("D : %d \n", count D);
  printf ("F : %d \n", count F);
  return 0;
}
```

b)

```
#include <stdio.h>
 int main (){
int arr [100];
int n, i; scan
  for (i=0; i
  scanf ("%d", &n);
```

```
for (i=0; i<n; i++)
{
  scanf ("%d", &arr[i]);
}
printf ("input elements :\n");
for (i=0; i<n; i++)
{
  printf ("%d ", arr[i]);
}
printf ("The reversed array :");
for (i= n-1; i>=0; i--)
{
  printf ("%d", arr[i]);
}
return 0;
}
```

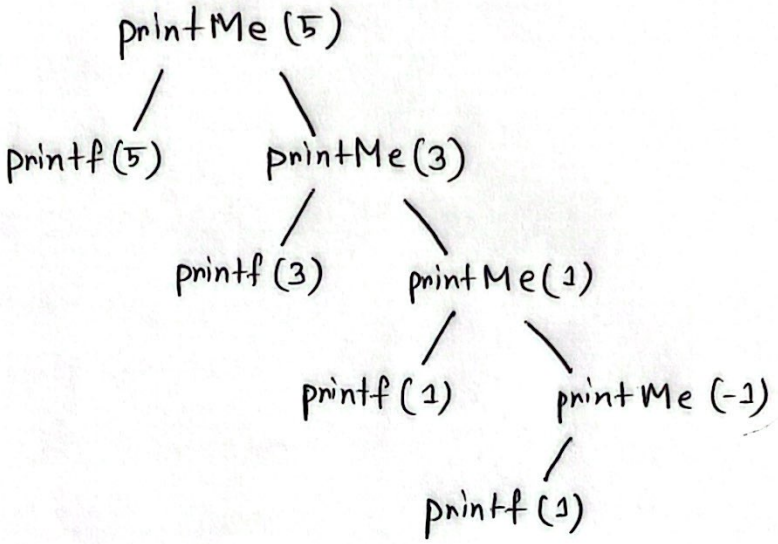
Ans. to the Q no. 5

a)

```
int a[] = {2,3,0,1,9,5};
int i = 2, b, c;
b = a[i] + a[i+1]; => 0+1 = 1;
c = a[a[i]]; => 2;
```

(Ans.)

b) The printMe function prints the value of n, calls itself with n-2 and then prints n again until the value of n becomes less or equal 0. So the block diagraw would be,



Ans. to the Q no. 5(c)

```
#include <stdio.h>
int main ()
{
  int n;
  scanf ("%d", &n);
  char filename [500];
  scanf ("%[^\n]", filename);
  FILE* file = fopen (filename, "w");
  for (int i = 2 ; i <= n ; i = i+2)
   {
    fprintf (file, "%d\n", i); }
```

```
    }
    fclose (file);
return 0;
}
```

Ans. to the Q no. 6

```
#include <stdio.h>
struct student {
    int ID;
    char gender;
    float class Test score;
    float mid Term score;
    float finalExamScore;
    float totalscore;
};
int countFails (struct student students [], int N) {
    int count = 0;
    for (int i = 0; i < N; i++)
    {
        if (students[i].totalscore < 40.0)
        {
            count++;
        }
    }
return count;
}
```

```
int Highest scorer (struct student students[], int N){
    int ~~top 5~~ Highestscored = 0;
    float HighestScore = 0;
  for (int i=0; i< N; i++)
  {
    if (students[i]. total score > Highest ~~max~~ score)
    {
      Highest score = students[i]. total score;
      Highest scored = student[i]. ID;
    }
  }
 return ~~the~~ Highest scored;
}
int main () {
  int N;
  scanf("%d", &N);
  struct Student students[N];
for (int i =0; i < N; i++)
 {
   scanf("%d", & students[i].ID);
   scanf ("%c", & students[i] . Gender);
   scanf ("%f". & students[i]. class Test score);
   scanf ("%f", & students [i]. Mid Term score);
   scanf ("%f", & students [i]. Final Exam score);
```

```
    students[i].totalscore = students[i].classTestscore +
    students[i].midTermscore + students[i].finalExamscore;
}
  int failed = countFails(students, N);
    printf("Number of failed students : %d", failed);

  int topscorerID = Highest scorer(students, N);
    printf("The ID of the topscorere : %d", topscorerID);

return 0;
}
```

b) Using 'fopen' to open a file might fail for various reasons. If the file is in a write mode then it might fail if the file does not have writing permissions or the program does not have the permission to write the directory, e.g. if the Disk space is full, if the file is already in use and more reasons might cause 'fopen' to fail. If the file is in a read mode then it might fail if the specified file doesn't exist, if it has permission issues, if the file is already in use and more reasons may cause 'fopen' to fail.

Ans. to the Q no. 6 (or)

a)

```
#include <stdio.h>
struct Movie star {
    char Name[50];
    int age;
    float annual earning;
    char gender;
};

float Lowest income (struct Movie star stars[], int N)
{
    float lowest income = ~~no~~ stars[0].annual earning;
    for (int i = 1; i < N; i++) {
        if (stars[i].annual earning < lowest income) {
            lowest income = stars[i].annual earning;
        }
    }
return lowest income;
}

int Oldest (struct Movie star stars[], int N)
{
    int Oldest = stars[0].age;
    for (int i = 1; i < N; i++)
    {
        if (stars[i].age > oldest)
        {
            oldest = stars[i].age;
        }
    }
```

```
return oldestage;
}
int main () {
    int N;
    scanf ("%d", &N);
    struct Moviestar stars [N];
  for (int i = 0; i < N; i++)
   {
    scanf (" %. [^\n]", stars [i]. name);
    scanf ("%. d", stars [i] . age);
    scanf ("%. f ", stars [i]. annual earnings);
    scanf ("%.c", stars [i]. gender);
   }
   float lowest income = LowestIncome (stars , N);
   printf ("The actor with the lowest income is : %.ft",
                                          lowest income)
   int oldestage = Oldest (stars, N);
   printf ("Age of the oldest movie stan : %.d \n", oldest
                                                    age);
  return 0;
}
```

b) when opening a file using 'fopen' we can open it in,

i) "r" : Read Mode;

ii) "w" ; Write Mode;

iii) "a" : append Mode;

iv) "rt" : read and write mode;

v) "wt" : write and read mode;

vi) "at" : Append and read mode;

Ans. to the Q no.1

a)

```
void main (){
 char s [80], t [80];
 gets (s);
 gets (t);
 if (strcmp (s,t) == strcmp (t,s))
     printf ("Yes");
  else
     printf ("No");
 }
```

if the input of two strings are "abc" & "abc123" the output will be "No". This is because the comparison strcmp (s1, s2) == strcmp (s2, s1) checks if the strings are equal in both directions and in this case they are not equal so the answer is NO.

b)

```
#include <stdio.h>
 int Vowel (char ch) {
  return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'
        || ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U');
 }
 int main ()
 {
   char sen [100];
   int vowels = 0;
   char consonant;
```

```
    scanf (" %[^\n]", sentence);
    for(int i = 0; sentence[i] != '\0'; i++)
    {
      if ((sentence[i] >= 'a' && sentence[i] <= 'z') ||
          (sentence[i] >= 'A' && sentence[i] <= 'Z'))
      {
        if (first consonant == '\0' && Vowel (sentence[i]))
        {
          consonant = sentence[i];
        }
        if (is Vowel (sentence[i])){
          vowels++;
        }
      }
    }
    printf ("First consonant : %c\n", cont consonant);
    printf ("Number of vowels : %d\n", vowels);

  return 0;
}
```

Ans. to the Q no. 1 (or)

So the use of 'strncpy' is to copy the string from one variable to another till 'n' numbers. Since we copied s till 4 strings and copied onto t. So the output will "Helld".

b)

```
#include <stdio.h>
int main() {
char sen1[100], sen2[100];
int vowels1=0, consonants1=0, Vowels2=0, consonants2=
0;
scanf ("%[^\n], sen1);
scanf ("%[^\n], sen2);
for (int i = 0 ; sen1[i] != '\0'; i++)
{
  if ((sen1[i] >='a' && sen1[i] <='z') || (sen1[i] >='A'
      && sen[i] <= 'Z'))
    {
    if (sen1[i] == 'a' || sen1[i] == 'e' || sen1[i]=='i'
      || sen1[i] == 'o' || sen1[i] == 'u' || sen1[i]=='A'
      || sen1[i] == 'E' || sen1[i] == 'I' || sen1[i]=='O'
      || sen1[i] == 'U')
      {
        vowels1++;
      }
    else
      {
        consonants1++;
      }
    }
  }
for (int i = 0 ; sen2[i] != '\0'; i++)
  {
```

b)

```
#include <stdio.h>
int countVowels (
int main () {
char sen1[100], sen2[100];
int vowels1=0; consonants1=0; Vowels2=0, consonants2=
                                        0;
scanf (" %[^\n], sen1);
scanf (" %[^\n], sen2);
for (int i = 0 ; sen1[i] != '\0'; i++)
{
  if ((sen1[i] >='a' && sen1[i] <='z') || (sen1[i] >='A'
      && sen[i] <= 'Z'))
    {
    if (sen1[i] == 'a' || sen1[i] == 'e' || sen1[i]=='i'
      || sen1[i] == 'o' || sen1[i] == 'u' || sen1[i]=='A'
      || sen1[i] == 'E' || sen1[i] == 'I' || sen1[i]=='O'
      || sen1[i] == 'U')
      {
       vowels1++;
       }
     else
       {
        consonants1++;
       }
    }
  }
for (int i = 0 ; sen2[i] != '\0'; i++)
  {
```

```
if ((sen2[i]>='a' && sen2[i]<='z') || (sen2[i]>='A'
    && sen2[i]<='Z'))
    {
     if (sen2[i]=='a' || sen2[i]=='e' || sen2[i]=='o'
        || sen2[i]=='i' || sen2[i]=='u' || sen2[i]=='A'
        || sen2[i]=='E' || sen2[i]=='I' || sen2[i]=='O'
        || sen2[i]=='U')
        {
          vowels2++;
        }
      else
        {
          consonahts2++;
        }
    }
}

 if (vowels1 == vowels2 && consonants1 == consonants2)
 {
   printf("YES");
 }
 else
    {
     printf("No");
    }
 return 0;
}
```

Ans. to the Q no. 2

a)
```
#include <stdio.h>
struct Employee {
    int employeeID;
    float salary;
};
float Maxsalary (struct Employee employees[], int N)
{
  float maxsalary = employees[0].salary;
  for (int i=1; i<N; i++)
  {
    if (employees[i].salary > maxsalary)
    {
      maxsalary = employees[i].salary;
    }
  }
return maxsalary;
}
int main()
{
  int N;
  scanf("%d", &N);
  struct Employee employees[N];
  for (int i=0; i<N; i++)
  {
```

```
        scanf ("%d", &employees [i]. employee ID);
        scanf ("%d", &employees [i]. salary );
    }
    float maxsalary = find ma MaxSalary (employees, N);
    printf ("Maximum salary among employees: .%.2f \n",
                                        maxsalary);
    printf ("size of the employee structure %d", sizeof
                                        (struct Employee));
    return 0;
}
```

Ord)

```
#include <stdio.h>
  struct MovieStar {
    int star ID;
    char name [50];
    int age;
};
float find Avg.age (struct Moviestar stars [ ], int N)
 {  int total age = 0;
    for (int i = 0; i < N; i++)
    {
      totalAge += stars [i]. age;
    }
return (float) totalAge / N;
}
```

```
int main ()
{
  int N;
  scanf ("%.d", &N);
  struct Moviestar stars [N];
  for (int i = 0; i < N; i++)
  {
    scanf ("%.d", & stars [i]. starID);
    scanf ("%.ds", stars [i]. name);
    scanf ("%.d", & stars [i]. age);
  }
  float averageAge = findAvg.age (stars, N);
  printf ("Average age among movie stars : %.2f\n",
                                        averageAge);
  printf ("size of the structure : %.lu \n", size of (struct
                                        Moviestar));
  return 0;
}
```

Ans. to the Q$_n$ no.3

a) The value of b, c and d after execution:

b = ++a [i]; ⇒ 7;

c = a [i++]; ⇒ 7;

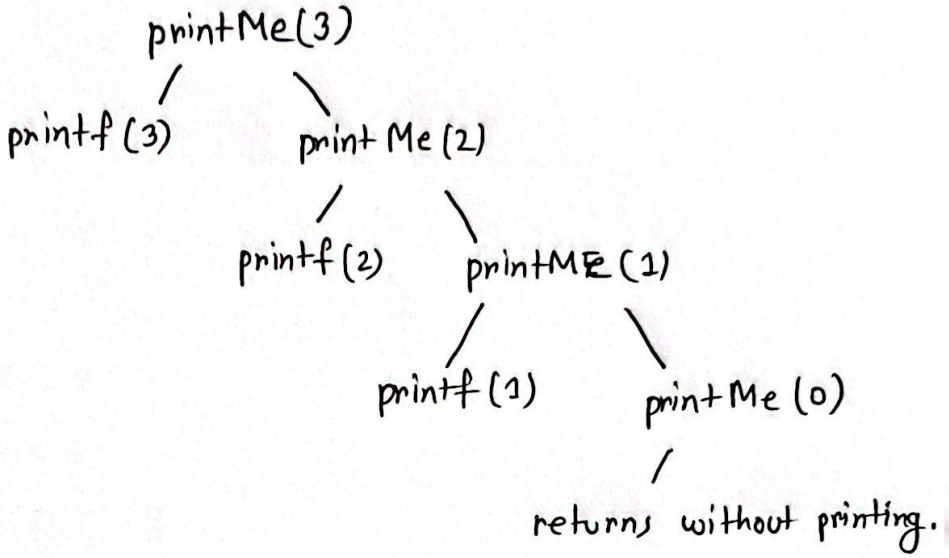
d = a[i] ⇒ 1;

b)
```
#include <stdio.h>
int main () {
  int for [100]; rev[
  int i, n;
  scanf ("%d", &n);
  for (i=0; i<n; i++)
  {
    printf ("
    scanf ("%d ", &for [i]);
  }
  for( i=0; i<n; i++)
  {
    printf ("%d", a[i]);
  }
  for (i=n-1 ; i>=0; i--)
  {
    printf ("%d ", a[i]);
  }
return 0;
}
```

Ans. to the Q no. 4

The 'printMe' function is a recursive function that prints numbers in a specific order. Let's analyze what happens whe 'printMe(3)' is called:

The block diagram will be,

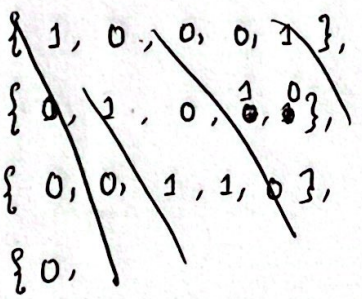


b)
```
#include <stdio.h>
int main (){
int a1, a2, a3;
scanf("%d %d %d", &a1, &a2, &a3);
if (a1 == 0 || a2 == 0 || a3 == 0)
 {
  printf ("The triangle is not valid \n");
 }
else
 {
  if (a1+a2+a3 == 180)
   {
    printf ("The triangle is valid \n");
   }
  else
    {
     printf ("The triangle is not valid \n");
    }
}
```

```
return 0;
}
```

Ans. to the Q no. 5 (a)

```
#include <stdio.h>
int main () {
int rows = 5;
int cols = 5;
int arr [5][6] = {
```



```
{1, 0, 0, 0, 1},
{0, 1, 0, 1, 0},
{0, 0, 1, 1, 0},
{0,
```

```
{1, 0, 0, 0, 0, 1},
{0, 1, 0 0, 1, 0},
{0, 0, 1, 1, 0, 0},
{0, 1, 0, 0, 1, 0},
{1, 0, 0, 0, 0, 1}
```

```
};
for (int i = 0; i < rows ; i++)
{
for (int j = 0; j < cols ; j++)
{
printf ("%d", arr [i] [j]);
}
printf ("\n");
}
return 0;
}
```

b)
```
#include <stdio.h>
int main () {
int rows, cols;
scanf("%d %d", &rows, &cols);
int arr [rows] [cols];
for (int i = 0; i < rows; i++)
  {
   for (int j = 0; j < ~~clos~~ cols; j++)
    {
      scanf ("%d", & array [i] [j]);
    }
  }
 for (int i = 0; i < rows; i++)
  {
    for (int j=0; j< cols; j++)
    {
     ~~scanf~~ printf ("%d", arr [i] [j]);
    }
    printf ("\n");
  }
int sum = 0;
for (int j =0; j < cols; j++) {
    int maxElement = arr [0] [j];
    for (int i = 1; i < rows; i++) {
      if (arr [i] [j] > maxElement)
         max Element = arr [i] [j];
      }
    }
    sum += max Element;                                    return 0;
}
printf ("Sum of maximum elements at column : %d \n", sum);
}
```

Ans. to the Q no. 6

a) if the address of a is 500 and address of b is 600;

	a	b	a_p	b_p
int a=5, b =0;	5	0	-	-
int *a_p= &a, *b_p= &b;	5	0	500	600
b = a+* b_p;	5	5	500	600
b_p = a_p;	5	5	500	500
a = (*a_p)* (*b_p);	25	5	500	500
*b_p = a/b;	25	5	500	500
* a_p = a%. b;	0	5	500	500

b) int x[5] = {10, 3, 7, 98, 7};

int *p;

p = &x[2];

i) p+2 : 702 +2* size of (int) = 708;

ii) *p+2 : 9;

iii) *(p+2) : 7;

iv) *(p+2)-*p : 0;

v) *(p) : 8;

c) When opening a file using 'fopen' in C, various emors can occur. Some common errors include:

1. If the specified file cannot be found, 'fopen returns NULL. This can cause errors.

2. If the program doesn't have the necessary permissions to access the file in the specified mode (read, write, execute), fopen may return NULL.

3. If there are not enough system resources (like memory) to open the file, fopen might fail.

4. If the file is already open by another process or by the same process in a conflicting mode, fopen might return NULL.

These are the reasons why 'fopen' might fail.