B.Tech First Year: Regular Course Lecture Plan Session 2022-23

Subject Name	Programming for Problem Solving

Unit No.	Unit Name	Syllabus Topics	Lecture No
4	Arrays & Basic Algorithms: Arrays:	Array notation and representation, manipulating array elements	23
		Using multi dimensional arrays	24
		Functions with array Passing arrays to functions	25
		Character arrays and strings	26
		Structure & Union	27
		Enumerated data types	28
	Basic Algorithms:	Searching : Linear Search	29
		Binary Search	30
		Basic Sorting Algorithms : Bubble Sort	31
		Selection Sort	32
		Insertion Sort, Notion of order of complexity	33
5	Pointer& File Handling: Pointers:	Introduction; declaration, applications of pointer	34
		Introduction to dynamic memory allocation:- malloc, calloc, realloc and free	35
		Linked List: Use of pointers in self-referential structures notion of linked list (no implementation)	36
	File handling:	File I/O functions	37
		File Programs	38
		Standard C preprocessors, Types of Preprocessor Directives	39
		Command-line arguments	40

Signature	
Name of Subject Head	Ms. Radhika Jindal

Unit: 5

Qu1- What is pointer and how is it initialized? (2015-16, 2016-17) (2017-18)

Ans Pointer is a special variable that stores the address of another variable. The pointer variable might be belonging to any of the data type (int, char, float).

Syntax of declaration of pointer:

data-type * variable name.

Ex:- int *p; char *q;

where * denotes that p is a pointer not a normal variable.

Declaration & initialization of pointer:

int a; int *p = &a;

Dereferencing a pointer: when * is used with the pointer variable then it is known as dereferencing a pointer when we dereference a pointer, then the value of the variable pointed by this pointer will be returned.

Note: The size of any pointer is compiler dependent. (2 byte for 16 bit compiler. Mostly equal to size of int)

Qu2- What is double pointer? How will you declare double pointer in C?

Ans Double pointer (pointer to pointer):- when we define a pointer to pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the adress of first pointer. That's why

Lecture No: 41

second pointer is called double pointer.

Declaration: Declaring double pointer is similar to declaring a pointer in C. The difference is we have to place an additional * before the name of the pointer.

Syntax:- data-type ** variable name.

Example:- int **p;

* 'C' Program to demonstrate pointer & double pointer.

```
# include <stdio.h>
int main()
{
  int var, *ptr1, **ptr2;
  var = 25;
  ptr1 = &var;
  ptr2 = &ptr1;
  printf("value of var = %d", var);
  printf("value of var using single pointer = %d", *ptr1);
  printf("Address of var using single pointer = %u", ptr1);
  printf("Address of ptr1 using double pointer = %u", *ptr2);
  printf("value of var using double pointer = %d", **ptr2);
  return 0;
}
```

Suppose address of var, ptr1 and ptr2 are 1000, 2000, 3000.

Then

1000	2000	3000
25	1000	2000
var	ptr1	ptr2

output: value of var = 25
value of var using single pointer = 25
Address of var using single pointer = 1000
Address of ptr1 using double pointer = 2000
value of var using double pointer = 25

V.imp
Qu3- Explain the types of pointer? Explain what is special about void pointer. How is it different from other pointers? (2015-16) Even/Odd

Ans There are different type of pointer which are as follows:-

1. NULL pointer
2. Void pointer
3. Wild pointer
4. Dangling pointer

1. NULL pointer: We create a NULL pointer by assigning the NULL. This method is used when you do not assign any address to the pointer. A NULL pointer always contain 0 value.

Ex: int *ptr = NULL;
printf("value inside ptr = %d", ptr);

O/P - value inside ptr = 0.

2. Void pointer: It is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be type cast to any type. It is created by using the keyword void. Pointer arithmetic is not possible of void pointer due to its concrete size. It can not be dereference.

Ex:- void *p, *q, *r; int a; float b; char *c;
p = &a; q = &b; r = &c; // void pointer can store address of any data type.

Que 7:- WAP to find sum of n elements using dynamic memory allocation.

Sol:-
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int n, i, *p, sum=0;
  printf("Enter how many numbers");
  scanf("%d", &n);
  p=(int*)malloc(n*sizeof(int));
                    //p=(int*)calloc(n, sizeof(int));
  if(p==NULL)
{
  printf("Memory not allocated");
  exit(0);
}
printf("Enter elements of array");
for(i=0; i<n; i++)
{ printf("Enter value");
  scanf("%d", p+i);
  sum = sum + *(p+i);
}
  printf("sum = %d", sum);
  return 0;
}
```


Que 8:- WAP to find the largest no. among 20 integers array using dynamic memory allocation.

Sol:-
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int *ar, max, i;
  ar = (int*) calloc(20, sizeof(int));
  if(ar==NULL)
    {
      printf("Memory not allocated");
      exit(0);
    }
printf("Input elements");
for(i=0; i<=19; i++)
  scanf("%d", ar+i);
printf("largest: ");
max = *ar;
for(i=1; i<=19; i++)
  { if(*(ar+i) > max)
      max = *(ar+i);
  }
printf("%d", max);
return 0;
}
```

Ques- What is life time of variable which is created dynamically?

Ans Dynamically allocated memory has (2018-19) no scope. It stays allocated until it is explicitly deallocated using free function.

V.imp

Ques Differentiate – (i) malloc() & calloc()
(ii) static & dynamic memory allocation.

(i)

malloc()	calloc()
① malloc() stands for memory allocation	① calloc stands for contigous allocation.
② malloc takes one argument i.e no. of bytes	② It takes two argument there are - no. of blocks & size of each block.
③ Syntax of malloc(): void * malloc(size_t n); allocates n bytes of memory if the allocation succeeds, a void pointer to the allocated memory is returned, otherwise NULL is returned.	③ syntax of calloc(): void * calloc(size_t n, size_t, size); Allocates a contiguous block of memory large enough to hold n elements of size bytes each. The allocated region is initialized to zero.
④ malloc is faster than calloc.	④ calloc takes little longer than malloc because of the extra steps of initializing the allocated memory by zero.

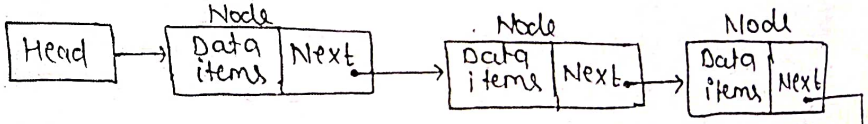
(ii)

Static memory allocation	Dynamic memory Allocation
① It is done before program execution (compile time)	① It is done during program execution (run time)
② It uses stack for managing static allocation of memory	② It uses heap for managing the dynamic allocation of memory.
③ It is less efficient (extra or less memory sometimes)	③ It is more efficient.
④ once the memory is allocated the memory size can't change	④ Memory size is flexible.
⑤ Execution is faster than dynamic memory allocation	⑤ Execution is slower than static memory allocation.
⑥ We can't reuse the unused memory.	⑥ The user can allocate more memory when required, also the user can release the memory when the user needs.
Example:- Array	Example:- Linked list

imp.

Ques What is linked list? Write the self reference structure of a node in linked list? Write the advantage of use of linked list. (2018-19)

Ans A linked list is a list of elements, which are connected together via link.



- Linked list can be visualized as a chain of nodes where every node points to next node.
- Linked list contains a head node have address of first node
- Each node carries a data field and a link field called next.
- Each node is linked with its next node using its next field

- last node carries a link as NULL to mark the end of the list.

Code for node:

```
struct node
{
  int data;
  struct node * next;
};
```

Advantage of linked list: (2020-21)

(1) Dynamic data structure:- A linked list is a dynamic arrangement so it can grow or shrink at running.

(2) No memory wastage:- The size of the linked list increase or decrease at run time so no memory wastage.

(3) Implementation: Linear data structure like stack & queue are easily implemented using a linked list.

(4) Insertion and deletion operation: Insertion & deletion operations are easier in linked list because the address present in next pointer needs to be updated.

Disadvantage of linked list:

(1) More memory usage: more memory required because pointer is also required to store the address of next element.

(2) Traversal is slow:- Traversal is more time consuming because accessing a node at position one has to traverse all the nodes before it.

(3) Reverse traversing: Not possible in singly linked list
(1) Possible in doubly linked list but needs extra memory for back pointer.

Type of linked list: (1) Singly
(2) Doubly
(3) circular

Q11. What is self referential structure?

Ans There are those structures that have one or more pointers which point to the same type of structure as their member.

Ex:-
```
struct node
{
  int data1;
  char data2;
  struct node * link;
};
```

In the above example 'link' is a pointer to a structure of type 'node'. Hence the structure 'node' is a self referential structure with 'link' as the referencing pointer.

Q12. What is file? what are the various type of files that can be created in C language. (2015-16, 2016-17, 2020-21)

Ans A file is a collection of bytes stored on a secondary storage device. The collection of bytes may be interpreted Ex: As character, words, lines, paragraph, pages from a textual document, fields and records belonging to a

data base or pixel from a graphical image.

FILE is a inbuilt structure in C

Declaration of file pointer:

```
FILE *f1;
```

- Opening of file:

```
f1 = fopen("file name", "mode");
```

The fopen() function is used to create a new file or to open an existing file.

- Closing of file:- `fclose(f1);`

fclose() function is used to close a file.

- To check existence of file:

```
if (f1 == NULL)
{ printf("File does not exists");
  exit(0);
}
```

Q: What is file handling? what is file operations in C?

Ans In C programming language the program store results, and other data of the program to a file using file handling in C. Also we fetch data from a file to work with it in the program. The operations that you can perform on a files in C are :-

File operations:
- creating a new file
- opening an existing file
- Reading data from existing file
- writing data to a file
- moving to a specific location on the file
- closing file
- Delete a file.

Q: How binary files differ from text files? (2015-16)

Ans There are three main differences between text and binary files.

1. Handling of new lines:- In text mode a new line character is converted into the carriage return-line feed combination before being written to the disk. In binary mode no such conversion takes place.
2. Representation of end of files: In text mode a special character is inserted after the last character in the file to marks the end of file. There is no such special character present in the binary mode. The binary file keeps track of the end of the file from the number of character present in the directory entry of the file.
3. Storage of numbers: storage of numbers in text file is inefficient as numbers are stored as string of characters in text mode.

imp.

Q: Give different modes in which these files can be used with proper syntax. (2015-16, 2017-18, 2019-20, 2020-21)

Ans
```
FILE *p;
p = fopen("filename", "mode");
```

There are 12 main types of file opening mode;

1. "r" :- Open file for reading and file must exists.
2. "w" :- Open file for writing. If file does not exists it created or if file already exist it is content is erased.
3. "a" :- Open file for appending. It adds all information at the end of the file leaving old data untouched. If file

r → read
w → write

does not exist it is created.

④ "r+" :- Open file for reading and writing and file must exist.

⑤ "w+" :- Open file for writing and reading. If file does not exist it is created or if file already exist its content is erased.

⑥ "a+" :- Open file for appending and reading. Again all data is written at the end of file leaving old data untouched. If file does not exists it is created.

The above 6 modes for text files then if you want to perform operations on binary files then you have to append b at the last. So six binary modes are :- "rb", "wb", "ab", "r+b", "w+b" & "a+b".

V. imp.

Q.11- Write and explain various input/output file function with example? (2015-16, 2017-18, 2020-21)

Ans ① fputc() or putc() :- These function write a character into a file. And after that increments file pointer.

Ex:- fputc(ch, fp); putc(ch, fp);

where ch - character value and fp - file pointer.

② fgetc() or getc() :- These functions read a character from a file pointed by fp. And after that increment file pointer.

Ex:- ch = fgetc(fp); ch = getc(fp);

where ch - character variable

fp - file pointer

③ putw() :- It is used to write an integer into a file. And after that increments file pointer.

Ex:- putw(i, fp);

where i - integer fp - file pointer

④ getw() :- It reads an integer from a file pointed by fp. And after that increment file pointer.

Ex:- i = getw(fp);

where i - integer fp - file pointer

⑤ fscanf() :- It is used to read formatted data from a file. In a C program we use fscanf() as below fscanf(fp, "%d%f", &age, &marks);

⑥ fprintf() :- It is used to write formatted data into a file.

fprintf(fp, "%s%d", var1, var2);

var1 :- string variable var2 :- integer variable

⑦ fread() :- This function reads set of data from the file.

Ex:- fread(&s, sizeof(s), 1, fp);

s - structure variable

⑧ fwrite() :- This function writes set of data into the file.

Ex:- fwrite(&s, sizeof(s), 1, fp);

Q.12- Define following functions

(i) remove() (ii) rewind() (iii) ftell()

Ans (i) remove():- This function deletes a file.

Ex: remove("file name");

(ii) rewind():. This function moves file pointer to the begining of the file

Ex:- rewind(fp);

(iii) ftell(): This function is used to get current position of a file pointer

Ex:- ftell(fp);

Ques: what is the use of fseek() in file? write its syntax. (2017-18)

Ans fseek() function is used to move file pointer position to the given location.

Syntax:- fseek(fp, long int offset, int whence);

where fp:- file pointer

offset:- Number of bytes/characters to be moved from whence the current file pointer position.

whence:- This is current file pointer position from where offset is added. Below 3 constants are used to specify this :-

SEEK_SET :- It moves file pointer position to the beginning of file.

SEEK_CUR :- It moves file pointer position to given location.

SEEK_END :- It moves file pointer position to the end of file.

CALL WAP in C to append some text at the end of an existed text file (2019-20)

Sol:

```
#include<stdio.h>
int main()
{
  FILE *p;
  char t;
  p = fopen("a.txt", "a");
  if (p==NULL)
  {
   printf("File can't open");
    exit(0);
  }
  printf("enter some text and $ for termination");
  while(1)
  {
   scanf("%c", &t);
    if(t=='$')
     break;
    fputc(t, p);
  }
   fclose(p);
   return 0;
  }
```

imp.

Class-A file named data.txt contains a series of integer numbers. WAP to read number from file and then write all "odd" number to file odd.txt and all "even" number to file even.txt (2019-20)

Ans

```
#include<stdio.h>
int main()
{
  FILE *fp, *fo, *fe;
  int n;
  fp = fopen("data.txt", "r");
  fo = fopen("odd.txt", "w");
  fe = fopen("even.txt", "w");
if((fp==NULL)||(fo==NULL)||(fe==NULL))
{
printf("File can't open");
  exit(0);
}
while((n=getw(fp))!=EOF)
 {
   if(n%2==0)
   putw(n,fe);
   else
    putw(n,fo);
 }
  fclose(fp);
  fclose(fo);
  fclose(fe);
   return 0;
 }
```


Ques: Suppose a file contains students records containing name and age of a student. WAP to read these records and display them in sorted order by name. (2015-16)

Ans

```
#include<stdio.h>
#include<string.h>
  struct student
  {
    int age;
    char name[50];
   };
    int main()
   {
    FILE *p;
    struct student s[50], t;
    int i=0, n=0, j;
    p=fopen("a.txt","r");
    if(p==NULL)
    {
      printf("File can't open");
       exit(0);
    }
    while(fscanf(p,"%d%s",&s[i],age,s
                          s[i].name)!=EOF)
    {
      i++;
      n++;
    for(i=0; i<n-1; i++)
    {
```

Lecture No: 45

```
for( j=0 ; j<n-1-i ; j++)
  {
    if( strcmp( s[j].name, s[j+1].name) >0 )
      {
        t = s[j] ;
        s[j] = s[j+1] ;
         s[j+1] = t;
      }
  }
  }
    for(i=0 ; i< n; i++)
    printf(" %d \t %s \n", s[i].age, s[i].name);
    return 0;
  }
```

imp.
Q11 22-WAP in C to count the number of character in a file and copy the text of that file to another file.
(2015-16, 2016-17, 2017-18, 2019-20, 2020-21)

Ans
```
#include <process.h>
#include <stdio.h>
  int main()
  {
    char ch;
    FILE *f1, *f2;
     int n=0;
    f1 = fopen("abc.txt", "r");
```

```
f2 = fopen("myz.txt", "w");
if ((f1==NULL) || (f2==NULL))
  {
    printf("File can't open");
     exit(0);
    while ((ch=fgetc(f))!=EOF)
    {
       n++;
       fputc(ch, f2);
    }
     printf("number of characters in file = %d", n);
  fclose(f1);
  fclose(f2);
  return 0;
}
```

Q12. What is pre-processor? Explain its role? (2015-16, 2019-20)

Ans Before the source code passes through the compiler. It is examined by the processor for any processor directives. Preprocessor directives begin with "#" and don't require semicolon.

1. macro substitution directive
2. File inclusion directive
3. compiler control directive (conditional compilation)

Role:- It removes comment, white space and modifies source code according to preprocessor directives. It converts .c file to .i file.

Que 24. Explain conditional compilation in detail with example. (2017-18, 2018-19)

Ans Six directives are available to control conditional compilation. They delimit blocks of program text that are compiled only if a specified condition is true. The begining of the block of program text is marked by one of three directives:

```
#if
#ifdef
#ifndef
```

optionally an alternate block of text can be set aside with one of two directives:

```
#else
#elif
```

The end of block or alternative block is marked by #endif directive.

If the condition checked by #if, #ifdef or #ifndef is true then all lines between the matching #else or #elif and #endif directive are ignored. If the condition is false then the lines between the #if, #ifdef or #ifndef directive are ignored.

Ex:-
```
# define CODE
int main()
{
 printf("a");
 #ifdef CODE
  printf("b");
 #else
  printf("c");
 #endif
 printf("d");
}
```
O/P :- abd

Example 2:
```
#define CODE
void main()
{
 printf("a")
 #ifndef CODE
  printf("b");
 #else
  printf("c");
 #endif
 printf("d");
}
```
O/P:- acd

Example 3:
```
# define CODE 10
int main()
{
 print("a");
 # if CODE > 20
   print ("b")
 # else
     #if CODE > 15
       printf("c");
     #else
       printf("d");
     # endif
 # endif
  printf("e");
}
```
O/P - ade

V.imp

Q1) What do you mean by macro? Explain type of macro with example? (2016-17, 2017-18, 2018-19,)

Ans Macro: A macro is a fragment a code that is given a name. You can use the fragment of code in your program by using the name. macro is defined by # define directive.

① object like macro (simple macro)
② function like macro (macro with argument)

1) object like macro: The object like macro is an identifier that is replaced by value. It is widely used to represent numeric constants.

Ex:- PI 3.14 or #define PI 3.14

When we use PI in our program it is replaced with 3.14

PI - macro template 3.14 - macro expression.

Ex1
```
# Include <stdio.h>
# define PI 3.14
  int main( )
  {
    float radius, area;
    printf ("Enter radius");
    scanf ("%f", &radius);
    area = PI * radius * radius;
    printf ("Area = %2f", area);
     return 0;
  }
```

2) Function like macro:- You can also define macros that works like a function call, known as function-like macro. Ex:- #define circleArea(r) (3.14*r*r)

Every time the program encounters circle Area (argument), it is replaced by (3.14 * argument * argument).

Suppose we passed 5 as argument then it expands as below:

Circle Area (5) expands to (3.14 * 5 * 5)

Ex:
```
#include <stdio.h>
#define circleArea(r) (3.14*r*r)
int main()
{
    float radius, area;
    printf("Enter radius");
    scanf("%f", &radius);
    area = circleArea(radius);
    printf("Area = %.2f", area);
    return 0;
}
```

Qu26- How the macros are defined and call? (2015-16, 2017-18)

Ans Macro is defined by # define directive.

Ex:- # define pi 3.14

You can use macro in your program by using the name.

Qu27- Define #pragma and #undef in C. (2017-18, 2018-19)

Ans pragma is used to call a function before and after main() function in C program

```
#include <stdio.h>
  void function1();
  void function2();
 # pragma startup function1
 # pragma exit function2
   int main()
   {
    printf("Now we are in main function\n");
    return 0;
   }
    void function1()
   {
    printf("function1 is called before main function\n");
   }
    void function2()
   {
    printf("function2 is called just after end of main function");
   }
```

O/P:- Function 1 is called before main function. Now we are in main function.
Function 2 is called just after end of main function.

#undef :- This preprocessor directive is used to undefine macro.

Syntax:- `#undef macro_name`

Ex:- C program to illustrate the use of #undef in a program.

```
#include <stdio.h>
#define PI 3.14
```



```
# undef PI
  int main()
  {
    float a, r=5;
    a = PI * r * r;
    printf("Area = %f", a);
     return 0;
   }
```

O/p!- Error undefined PI

Qu28:-

```
#define PRODUCT(n) n*n
 int main()
 {
  int j;
  j=64/PRODUCT(4);
  printf("%d",j);
   return 0;
 }
```

what will be the output ? (2017-18)

Ans Preprocessor replace PRODUCT(4) with 4*4 so

```
j = 64/PRODUCT(4)
j = 64/4*4;
j = 64 * 16 * 4;
j = 64
```

so 64 will be print.

Que? Write macro difinition with argument for calculation to simple interest and amount. Store these macro difinition in a file called "interest.h" Include this file in program and use macro difinition for calculation simple intrest and amount. (2019-20)

Ans
```
#define amount(p,r,t) (interest(p,r,t)+p)
#define interest(p,r,t) (p*r*t)/100)

#include <stdio.h>
#include "interest.h"
  int main()
  {
    float p,r,t,s,a;
  printf("Enter money, rate of interest and year");
  scanf("%f %f %f", &p, &r, &t);
      s=interest(p,r,t);
      a=amount(p,r,t);
      printf("Interest =%f and amount = %f", s,a);
      return 0;
     }
```

V. imp.
Que What is command line argument in C? (2018-19)

Ans Command line argument is a parameter supplied to the program when it is invoked. Command line argument are passed to main() function.

Syntax:- int main(int argc, char * argv[])

Here argc is counts the number of arguments on the command line.

- argv is a pointer array which holds pointer of type char which points to the argument passed to the program.

Q13. WAP to print arguments passed by command line run as
C:/tc/test Hello I am here

Ans
```
#include <stdio.h>
int main (int argc, char *argv[])
{
  int i;
  for(i=0; i<argc; i++)
  printf ("%s", argv[i]);
  return 0;
}
```

O/P:- test Hello I am here

Q14. WAP to sum of n number using command line argument.

Ans
```
#include <stdio.h>
int main (int argc, char *argv[])
{
  int i, sum=0;
  for(i=0; i<argc; i++)
  {
    sum = sum + atoi(argv[i]);
  }
  printf("sum = %d", sum);
  return 0;
}
```

Output: sum = 25

B. Tech I Year [Subject Name: Programming for Prob. Sol.]

5 Years AKTU University Examination Questions		Unit-5	
S. No	Questions	Session	Lecture No
1	What is special about void pointer?	15-16 (Even)	34-40
2	What is pointer and how is it initiallized?	15-16(Even) 16-17(Odd) 17-18 (Odd)	34-40
3	What do you mean by pointer arithmetic?	19-20 (Odd)	34-40
4	What is void pointer? How is it different from other pointers?	15-16(Odd)	34-40
5	State the features of pointers. WAP to sort given numbers using pointers.	15-16 (Odd) 16-17 (Odd) 19-20(Odd)	34-40
6	What is the role of dynamic memory allocation?	15-16 (Odd)	34-40
7	What is life time of variable which is created dynamically?	18-19 (Odd)	34-40
8	Difference between compile time and run time memory allocation.	18-19 (Even)	34-40
9	What is dynamic memory allocation? How does it help in building complex programs? What is the task of following memory allocation function? a. malloc b. calloc c. free d. realloc	15-16 (Odd) 16-17 (Odd) 17-18 (Odd) 18-19 (Odd) 19-20 (Odd) 20-21(Odd)	34-40
10	What is linked list? Write the self reference structure of a node in linked list?	18-19 (Odd) 18-19 (Even)	34-40
11	Write the advantage of use of linked list.	20-21 (Odd)	34-40
12	How binary file differ from text file.	15-16 (Odd)	34-40
13	What is the use of fseek() function in files. Write its syntax.	17-18 (Even)	34-40
14	Write the following function in file operations: i. getw() ii. putw() iii. fscanf() iv. fprintf()	17-18 (Even) 20-21 (Odd)	34-40
15	Explain file handling.	20-21 (Odd)	34-40
16	What are the various types of files that can be created in C language? Also give different modes in which these files can be used with proper syntax.WAP in c to append some text at the end of an existed text file.	15-16 (Odd) 17-18 (Odd) 19-20 (Odd) 20-21 (Odd)	34-40
17	List various file operation in C .WAP in c to count the number of characters in a file.	15-16 (Odd) 15-16 (Even) 16-17 (Odd) 18-19 (Even)	34-40
18	Write the various input functions used in file handling in c. A file	15-16 (Odd)	34-40

	named data contains a series of integer numbers WAP to read number from file and then write all 'odd' number to file ODD.txt & all even to file EVEN.txt.	19-20 (Odd)	
19	Suppose a file contains student's records with each record containing name and age of a student. WAP in C to read these records and display them in sorted order by name.	15-16 (Even)	34-40
20	Write a c program to copy the text of one file to another.	15-16 (Odd) 16-17 (Odd) 17-18 (Odd) 20-21 (Odd)	34-40
21	How the macros are defined and called in c. explain with suitable example.	15-16 (Odd)	34-40
22	What is the role of C preprocessor?	15-16 (Odd)	34-40
23	Write the difference between #includes<stdio.h>& #include"stdio.h" file inclusion method.	15-16 (Even)	34-40
24	What is preprocessor directive? Explain any three of them.	15-16 (Even) 19-20 (Odd) 20-21 (Odd)	34-40
25	What do you mean by macro? Explain type of macros with examples.	16-17 (Odd) 17-18 (Odd) 18-19 (Odd) 18-19 (Even)	34-40

26	#define PRODUCT(n) n*n
	void main( )
	{
	int j;
	j=64/PRODUCT(4);
	printf("%d",j);
	}
	What will be the output of the above program?
		17-18 (Odd)	34-40

27	Explain the syntax and use of the following directive with example: i. #ifdef ii. #undef iii. #pragma iv. #include	17-18 (Even) 18-19 1-sem	34-40
28	Write macro definition with arguments for calculation of simple interest and amount. Store these macro definitions in a file called 'interest.h' . Include this file in your program and use macro definitions for calculation simple interest and amount.	19-20 1-sem	34-40
29	Explain command line argument in c with example.	18-19 1-sem	34-40