CS6202 PROGRAMMING AND DATA STRUCTURES I L T P C 3 0 0 3

OBJECTIVES:

· To introduce the basics of C programming language
· To introduce the concepts of ADTs
· To introduce the concepts of Hashing and Sorting

UNIT I C PROGRAMMING FUNDAMENTALS- A REVIEW 9

Conditional statements – Control statements – Functions – Arrays – Preprocessor - Pointers -
Variation in pointer declarations – Function Pointers – Function with Variable number of arguments

UNIT II C PROGRAMMING ADVANCED FEATURES 9

Structures and Unions - File handling concepts – File read – write – binary and Stdio - File
Manipulations

UNIT III LINEAR DATA STRUCTURES – LIST 9

Abstract Data Types (ADTs) – List ADT – array-based implementation – linked list implementation –
– singly linked lists- circularly linked lists- doubly-linked lists – applications of lists – Polynomial
Manipulation – All operation (Insertion, Deletion, Merge, Traversal)

UNIT IV LINEAR DATA STRUCTURES – STACKS, QUEUES 9

Stack ADT – Evaluating arithmetic expressions- other applications- Queue ADT – circular queue
implementation – Double ended Queues – applications of queues

UNIT V SORTING, SEARCHING AND HASH TECHNIQUES 9

Sorting algorithms: Insertion sort - Selection sort - Shell sort - Bubble sort - Quick sort - Merge sort -
Radix sort – Searching: Linear search –Binary Search Hashing: Hash Functions – Separate
Chaining – Open Addressing – Rehashing – Extendible Hashing
TOTAL: 45 PERIODS 9

TEXT BOOKS:

1. Brian W. Kernighan and Dennis M. Ritchie, “The C Programming Language”, 2nd Edition,
Pearson Education, 1988.
2. Mark Allen Weiss, “Data Structures and Algorithm Analysis in C”, 2nd Edition, Pearson
Education, 1997.

REFERENCES:

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest, Clifford Stein, “Introduction to
Algorithms", Second Edition, Mcgraw Hill, 2002.
2. Reema Thareja, “Data Structures Using C”, Oxford University Press, 2011
3. Aho, Hopcroft and Ullman, “Data Structures and Algorithms”, Pearson Education,1983.
4. Stephen G. Kochan, “Programming in C”, 3rd edition, Pearson Ed.,

Conditional statements – Control statements – Functions – Arrays – Preprocessor - Pointers -
Variation in pointer declarations – Function Pointers – Function with Variable number of arguments

1. **Differentiate between pointer to constants and constants pointers**

Constant Pointers : These type of pointers are the one which cannot change address they are pointing to. This means that suppose there is a pointer which points to a variable (or stores the address of that variable). If we try to point the pointer to some other variable (or try to make the pointer store address of some other variable), then constant pointers are incapable of this.
A constant pointer is declared as : 'int *const ptr' (the location of 'const' make the pointer 'ptr' as constant pointer)
Example:

```
#include<stdio.h>
int main(void)
{
   int a[] = {10,11};
   int* const ptr = a;
   *ptr = 11;
   printf("\n value at ptr is  : %d\n",*ptr);
   printf("\n Address pointed by ptr  : %p\n",(unsigned int*)ptr);
   ptr++;
   printf("\n Address pointed by ptr  : %p\n",(unsigned int*)ptr);
   return 0;
 }
```

Pointer to Constant : These type of pointers are the one which cannot change the value they are pointing to. This means they cannot change the value of the variable whose address they are holding.

A pointer to a constant is declared as : 'const int *ptr' (the location of 'const' makes the pointer 'ptr' as a pointer to constant.

Example:

```
#include<stdio.h>
int main(void)
{
   int a = 10;
   const int* ptr = &a;
   printf("\n value at ptr is  : %d\n",*ptr);
   printf("\n Address pointed by ptr  : %p\n",(unsigned int*)ptr);
   *ptr = 11;
   return 0;
}
```

2. **What is Sizeof operator in C?**

Sizeof is unary operator used to calculate the sizes of data types, this operator can be applied to all data types. The size of operator is used to determine the amount of memory space that the variable/expression/ data type will take.
For example,

```
int a=10, result
Result=sizeof (a);
```

Output:
result=2.
Since a is integer, it requires 2 bytes of storage space.

3. **Explain about Formatted and Unformatted I/O Functions.**

Unformatted I/O functions:
1.getchar():Used to read a character
2.putchar():Used to display a character
3.gets():Used to read a string
4.puts():Used to display a string which is passed as argument to the function

Formatted I/O functions:
printf() is an example of formatted output function and scanf() is an example of formatted input function.

4. **Compare While, Do-While and For Statement?**
   While Loop: It provides a mechanism to repeat one or more statements while a particular statement is true, in while loop first the condition is tested and then the block of statement is executed. It is an entry controlled loop Structure.
   Syntax:
   Statement x;
   While(condition)
   {
   Statement block;
   }
   Statement y;

   Do-while: This loop is similar to while loop, the only difference is that in do-while loop the condition is tested at the end of the loop. It is an exit controlled loop Structure.
   Syntax:
   Statement x;
   do
   {
   Statement block;
   } While(condition);
   Statement y;

   For loop: It used the loop variable is initialized only once. With every iteration of the loop, the value of loop variable is updated and the condition is checked. If the condition is true the statement block for the loop is executed else the control jumps to immediate statement following the For loop. It is an entry controlled loop Structure.
   Syntax:
   for(initialization;condition;increment/decrement/update)
   {
   Statement Block;
   }
   Statement y;
5. **What is type casting?**
   Type casting is the process of converting the value of an expression to a particular data type.
   Example:
   int x,y;
   c = (float) x/y; where a and y are defined as integers. Then the result of x/y is converted into float.

6. **Distinguish between Call by value Call by reference.**
   Call by value
   a) In call by value, the value of actual agreements is passed to the formal arguments and the operation is done on formal arguments.
   b) Formal arguments values are photocopies of actual arguments values.
   c) Changes made in formal arguments valued do not affect the actual arguments values.
   Call by reference.
   a) In call by reference, the address of actual argurment values is passed to formal argument values.
   b) Formal arguments values are pointers to the actual argument values.
   c) Since Address is passed, the changes made in the both arguments values are permanent

7. **What is the difference between an array and pointer?**
   Array
   1.Array allocates space automatically
   2.It cannot be resized.

3.It cannot be reassigned.
4.Size of(array name) gives the number of bytes occupied by the array.

Pointer

1.Pointer is explicitly assigned to point to an allocated space.
2.It can be resized using realloc ().
3.Pointers can be reassigned.
4.Sizeof(pointer name) returns the number of bytes used to store the pointer variable.

8. **How many storage classes does C language supports. Why we need these types of such classes? Explain each Type with an Example.**

Storage class of a variable defines the scope (visibility) and life time of variables. It is needed because it specifies how long the storage allocation will continue to exist for the function or variable. It specifies the scope of variable or function. It specify whether variable will automatically initialized to zero or in- determinate value. C supports four storage classes: automatic, register, external, and static.

Auto: The auto storage class specified is used to explicitly declare the variable with automatic storage. It is the default storage class for the variables declared in a block.
Eg: auto int x;

Register: When a variable is declared using register as it storage class, it is stored in CPU register instead of RAM. Since the variable in RAM, the maximum size of the variable is equal to the register size. It is used when a programmer need quick access to the variable.
Eg: Register int x;

External: It is used to give a reference of global variable that is visible to all the program files.
Eg: Extern int x;

Static: While auto is the default storage class for all local variables, static is the default storage class for global variables. Static variable have a life over the entire program. ie., memory for the static variables is allocated when the program begins running and is freed when the program terminates.
Eg: Static int x=10;

9. **Define function. Why they are needed? What are its types?**

A function is a group of statements that together perform a task. C enables its programmer to break-up a program into segment commonly known as function, each of which can be written more or less independently of the others. Every function in the program supposed to perform a well defined task. It is needed because, dividing the program into separate well defined functions facilitates each function to be written and tested separately. Understanding,coding and testing multiple separate function are easier than doing the same for one huge function.

Types:

1. Built in Functions
2. User Defined functions

10. **What is a Pointer and how it is declared? What are the uses of Pointers?**

Pointer is a variable which holds the address of another variable.

Pointer Declaration:

datatype *variable-name;

Example:

int *x, c=5;
x=&a;

Uses of Pointers

Pointers are used to return more than one value to the function
· Pointers are more efficient in handling the data in arrays
· Pointers reduce the length and complexity of the program
· They increase the execution speed
· The pointers save data storage space in memory

11. **Write short notes on null pointers, and generic pointers. How to declare and initialize these pointers?**

Null Pointer: It is a special pointer value that is known not to point anywhere, this menas that the null pointer does not to point to any valid memory address.
Eg: int * ptr=NULL;

Generic pointer: It is a pointer variable that has void as its data type. The void pointer, or the generic pointer, is a special tyoe of pointer that can be pointed at variable of any data type. For example, void * ptr;

12. What is an Array? What are its types? List out the operations performed on an array? Write down the general form of calculating length of the Array.

Array:

An array is a derived data type, which is a collection of element of same type stored in a physically continues memory location. Example, int marks [10];

Types of arrays:

1) Single dimensional array. eg: int a[5];
2) Multi dimensional array. eg: int a[5] [5];

List of Operations performed on array:

- Insertion: insert an element into the array.
- Deletion: Delete an element from the array.
- Traversal: Accesses each element of the array.
- Sort: Re-arranges the array element in a specific order.
- Search: Searches the key value in the array.

Calculating the length of the Array:

The length of the array is given by the number of elements stored in it. The general formula to calculate the length the array is, length = index of the last element – index of the first element +1.

13. Write a program to print the reverse of a string.

```
#include <stdio.h>
#include <conio.h>
# include <string.h>
void main()
{
char str[30], rev[30];
int i,len;
clrscr ();
Printf ("\n enter a string");
gets(str);
len=strlen(str);
for(i=0;i<len;i++)
rev[len-i-1]=str[i];
rev[len]='\0';
printf ("\n\n the reverse of string %s is %s",str, rev);
getch ();
}
```

14. Write a program to concatenate two strings.

```
#include <stdio.h>
#include <conio.h>
# include <string.h>
void main()
{
char str1[30], str2[30];
clrscr ();
Printf ("\n enter string 1:\n");
gets(str1);
Printf ("\n enter string2:\n");
gets(str2);
printf("the result of concatenation is:");
printf("%s",strcat(str1,str2,strlen(str2)));
getch();
}
```

15. **Write a program to find the greatest amongst two numbers using conditional / ternary operator.**

```
#include <stdio.h>
#include <conio.h>
int main()
{
int a,b,large;
clrscr ();
printf("\n enter the 3 numbers:");
scanf("%d%d",&a,&b);
large=a>b?a:b;
printf("\n the largest number is:%d",large);
return 0;
}
```

16. **Write a program to find the largest 3 numbers using function**.

```
#include <stdio.h>
int greater(int ,int ,int);              //function declaration
int main()
{
int a,b,c,large;
printf("\n enter the 3 numbers:");
scanf("%d%d%d",&a,&b,&c);
large=greater(a,b,c);                    //function call
printf("\n largest number=%d",large);
return 0;
}
int greater( int i,int j,int k)       //function definition
{
if(i>j && i>k)
return i;
if(j>i&&j>k)
return j;
else
return k;
}
```

17. **Is it better to use a macro or a function?**

Macros are more efficient (and faster) than function, because their corresponding code is inserted directly at the point where the macro is called. There is no overhead involved in using a macro like there is in placing a call to a function.

However, macros are generally small and cannot handle large, complex coding constructs. In cases where large, complex constructs are to handled, functions are more suited, additionally; macros are expanded inline, which means that the code is replicated for each occurrence of a macro.

18. **Write an example program for Macros with Arguments.**

```
#include<stdio.h>
#include<conio.h>
#define s(x) x*x
Void main()
{
Clrscr();
Printf("%d",s(5));
}
```

Output: 25

19. **Write an example program for Nesting of Macros.**

```
#include<stdio.h>
#include<conio.h>
#define s(x)   x*x
#define c(x)   x*s(x)
```

```
Void main()
{
Clrscr();
Printf("%d",c(4));
}
```
Output:4*4*4=64

20. What is conditional directive? How does it help a programmer?

A conditional directive is used instruct the preprocessor to select whether or not to include a chunk of code in the final token stream passed to the compiler. The preprocessor conditional directives can test arithmetic expressions, or whether a name is defined as a macro, etc. The if-else directives #if, #ifdef, #ifndef, #else, #elif and #endif can be used for conditional compilation.

Conditional preprocessor directives can be used in the following situations:

1. A program may need to use different code depending on the machine or operating system it is to run on.
2. The conditional preprocessor directive is very useful when you want to compile the same source file into two different programs. While one program might make frequent time-consuming consistency checks on its intermediate data, or print the values of those data for debugging, the other program, on the other hand can avoid such checks.
3. The conditional preprocessor directives can be used to exclude code from the program whose condition is always false but is needed to keep it as a sort of comment for future reference.

21. Distinguish between #ifdef and #if directives.

#ifdef

#ifdef is the simplest sort of conditional preprocessor directive and is used to check for the existence of macro definitions. Its syntax can be given as:

#ifdef *MACRO*
    controlled text
#endif

The #if Directive

The #if directive is used to control the compilation of portions of a source file. If the specified condition (after the #if) has a nonzero value, the controlled text immediately following the #if directive is retained in the translation unit. The #if directive in its simplest form consists of

#if condition
    controlled text
#endif

While using #if directive, make sure that each #if directive must be matched by a closing #endif directive. Any number of #elif directives can appear between the #if and #endif directives, but at most one #else directive is allowed. However, the #else directive (if present) must be the last directive before #endif.

22. Define the terms: File inclusion Directive and Pragma Directives.

File inclusion Directive:

When the preprocessor finds an #include directive it replaces it by the entire content of the specified file. There are two ways to specify a file to be included:

#include "file"
#include <file>

Pragma Directive:

The #pragma directive is a compiler-specific directive, used to control the actions of the compiler in a particular portion of a program without affecting the program as a whole. The effect of pragma will be applied from the point where it is included to the end of the compilation unit or until another pragma changes its status. A #pragma directive is an instruction to the compiler and is usually ignored during preprocessing.

The syntax of using a pragma directive can be given as:

#pragma *string.*

16 marks

1. **Explain briefly about Various Control Structures in C with suitable examples.**
2. **(i) Write a program to calculate the sum of numbers from m to n.**
   (ii) Write a program to find the greatest amongst three numbers using conditional operator.
3. **(i) Write a program to print the Fibonacci series using recursion.**
   (ii)Write a program to enter the marks of a student in 4 subjects and then calculate the total, average and display the grade obtained by the student.
4. **(i) Write a program to determine whether an entered character is vowel or not, using switch case.**
   (ii)Write a program using FOR loop to calculate the average of first n natural numbers.
5. **(i)Explain about passing parameters to the function with and without using pointers with suitable example.(or) Explain about Call by Value and Call by Reference with an example program.**
   (ii)What is Recursion? Explain Different types of recursion.
6. **What is a String? Explain about various String operations that are performed on character arrays with an example.**
7. **Explain in detail about various Pre-processor directives with suitable examples.**
8. **What is an Array? What are its types? Write a program for matrix multiplication.**
9. **What is a pointer? Explain about (i) pointer to functions ,(ii)passing a Function Pointer as an Arguments to a Function and (iii) Pointer to Pointers with an example.**
10. **(i)Write a program to insert a number at a given location in an array.**
    (ii)Write a Program to merge two unsorted Arrays.

UNIT II CPROGRAMMING ADVANCED FEATURES 9

Structures and Unions -File handling concepts –File read –write –binary and Stdio -File Manipulations

1. Compare arrays and structures.

Comparison of arrays and structures is as follows.

Arrays	**Structures**
An array is a collection of data items of same data type.Arrays can only be declared.	A structure is a collection of data items of different data types. Structures can be declared and defined.
There is no keyword for arrays.	The keyword for structures is struct.
An array name represents the address of the starting element.	A structrure name is known as tag. It is a Shorthand notation of the declaration.
An array cannot have bit fields.	A structure may contain bit fields.

2. Compare structures and unions.

Structure	**Union**
Every member has its own memory.	All members use the same memory.
The keyword used is struct.	The keyword used is union.
All members occupy separate memory location, hence different interpretations of the same memory location are not possible. Consumes more space compared to union.	Different interpretations for the same memory location are possible. Conservation of memory is possible

3. Define Structure in C.

C Structure is a collection of different data types which are grouped together and each element in a C structure is called member.

If you want to access structure members in C, structure variable should be declared.

Many structure variables can be declared for same structure and memory will be allocated for each separately.

It is a best practice to initialize a structure to null while declaring, if we don't assign any values to structure members.

4. What you meant by structure definition?

A structure type is usually defined near to the start of a file using a typedef statement. typedef defines and names a new type, allowing its use throughout the program. typedefs usually occur just after the #define and #include statements in a file.

Here is an example structure definition.

```
typedef struct { char

        name[64];

        char course[128];

        int age;

        int year;

} student;
```

This defines a new type student variables of type student can be declared as follows.

```
student st_rec;
```

5. How to Declare a members in Structure?

A **struct** in C programming language is a structured (record) type[1] that aggregates a fixed set of labeled objects, possibly of different types, into a single object. The syntax for a struct declaration in C is:

```
struct tag_name

{

  type
  attribute;
  type
  attribute2;
};
```

6. What is meant by Union in C.?

A **union** is a special data type available in C that enables you to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multi-purpose.

7. How to define a union in C.

To define a union, you must use the **union** statement in very similar was as you did while defining structure. The union statement defines a new data type, with more than one member for your program. The format of the union statement is as follows:

```
union [union tag]
{member definition;
  ...
  member definition;
} [one or more union variables];
```

8. How can you access the members of the Union?

To access any member of a union, we use the **member access operator (.)**. The member access operator is coded as a period between the union variable name and the union member that we wish to access. You would use **union** keyword to define variables of union type.

9. What are storage classes?

A storage class defines the scope (visibility) and life time of variables and/or functions within a C Program.

10. What are the storage classes available in C?

There are following storage classes which can be used in a C Program

Auto,register,static,extern

11..Write a note on File.

A file is a collection of related information normally representing programs, both source and object forms and data. Data may be numeric, alphabetic or alphanumeric. A file can also be defined as a sequence of bits, bytes, lines or records whose meaning is defined by the programmer. Operations such as create, open, read, write and close are performed on files.

12. Explain the terms field,record,file organization,key,and index.

A data field is an elementary unit that stores a single fact.

A record is a collection of related data fields that is sen as a single unit from the application

A File is a collection of related records.

A Key the part of data BY WHICH IT IS STORED,INDEXED,CROSS REFERENCE,ETC.

A Indax file that stores keys and index into another file.

13.What is File Handling:

We frequently use files for storing information which can be processed by our programs. In order to store information permanently and retrieve it we need to use files .A file is collection of related data .Placed on the disk.Files are not only used for data. Our programs are also stored in files.

14.Why we use file Handling:

The input and out operation that we have performed so far were done through screen and keyboard only . after the termination of program all the entered data is lost because primary memory is volatile . if the data has to be used later ,then it becomes necessary to keep it in permanent storage device. so the c language provide the concept of file through which data can be stored on the disk or secondary storage device . the stored data can be read whenever required.

15.what are Types of File Handling in C?

The file handling in c can be categorized in two types-

1. **High level (Standard files or stream oriented files)-** High level file handling is managed by library function. High level file handling commonly used because it is easier and hide most of the details from the programmer.
2. **Low level (system oriented files)-** low level files handling is managed by system calls.

16.Different Ways of storing data in files?

There are two ways of storing data in files.

Text Format-In text format data is stored as a line of character with each line terminated by a new line character ('\n'). Text files are in human readable form they can be created and read using any text editor.

Binary Format-- In binary format data is stored on the disk same way as it is represented in the computer memory . binary files are not in human readable form they can be created and read by a specific program written for them .The binary data stored in the file can-t be read by any editor.

The input and output operation in binary files take less time as compared to that of the text files because in binary files no conversion have to take place .However the data written using binary format is not very portable since the size of data types and byte order may be different on different machine. In text format , these problem do not arise. so it is more portable.

17.What are the Step for file operation in C programming?

1. Open a file
2. Read the file or write data in the file
3. Close the file

18.Define File modes?

File mode	Description
r	Open a text file for reading.
w	Create a text file for writing, if it exists, it is overwritten.
a	Open a text file and append text to the end of the file.
rb	Open a binary file for reading.
wb	Create a binary file for writing, if it exists, it is overwritten.
ab	Open a binary file and append data to the end of the file

19.Define End of file?

The file reading function need to know the end of file so that they can stop reading .when the end of file is reached , the operating signal send s an end of file signal to the program. when the program receive this signal , the file reading function return EOF which is constant defined in the file stdio.h an its value -1 . EOF is an integer value so make sure the return value of the function Ithe ares assigned to a integer variables.

20.what are the Function used for file Handling

1. Character I/O - fgetc( ), fputc( ), getc( ), putc( ).
2. String I/O - fgets( ), fputs( ).
3. Integer I/O - getw( ), putw( ).
4. Formatted I/O - fscanf( ), fprintf( ).
5. Record I/O - fread( ), fwrite( ).

21. Define Formatted I/O

C provide two type of formatted function.

1. fprintf( ) function
2. fscanf( ) function

fprintf( ) function: printf( ) function used for Write data to text file.

```
fprintf (FILE *fptr , const char  *format[argu1,argu2 .... argun  ]);
```

fscanf () function: fscanf function Reads data from the *stream* and stores them according to the parameter *format* into the locations pointed by the additional arguments. The additional arguments should point to already allocated objects of the type specified by their corresponding format tag within the *format* string.

```
fscanf (FILE *fptr , const char *format [,address,.....]);
```

22.Define Random Access to file in C

Random access means we can move to any part of a file and read or write data from it without having to read through the entire file. we can access the data stored in the file in two ways.

1. Sequentially
2. Randomly

23.How to use fseek() function in C?

This function is used for setting the file position pointer at the specified bytes . **fseek** is a function belonging to the ANCI C standard Library and included in the file stdio.h. Its purpose is to change the file position indicator for the specified stream.

```
int fseek(FILE *stream_pointer, long offset, int origin);
```

24. How to use ftell() Function in C ?

This function return the current position of the file position pointer. The value is counted from the beginning of the file.

```
long ftell (file * fptr);
```

25. What are the file attributes in C?

Read - Only allows a file to be read, but nothing can be written to the file.

Archive - Tells Windows Backup to backup the file.

System - System file.

Hidden - File will not be shown when doing a regular dir from DOS.

26.What are the Basic file Operations?

Creating a file
Updating a file
Retrieving from a file
Maintaining a file

27 .What are the types of file accessing?

a) Sequential access This type of file is to be accessed sequentially that is to access aparticular data all the preceding data items have to be read anddiscarded.

b) Random access

This type of file allows access to the specific data directly with outaccessing its preceding data items

PART B

1. **Write a Program using structure to read and display the information about an employee.**
2. **Write a program using pointer to structure to initialize the members in a structure.Use functions to print the employee's information.**
3. **Explain with an example how structures are initialized.**
4. **Write in detailed about formatted I/O?**
5. **Explain in detailed about Manipulators?**
6. **Explain the hierarchy of File stream classes?**
7. **Write about Random Access to a file?**
8. **Write a program to copy content of one file to another using command line arguments**
9. **Write a program to write text in a file. Read the text from the file from end of file. Display the contents of file in reverse order.**
10. **Write a program that swaps each character pair in a text file. Eg if a file contains "1234" then after swapping the file will contain "2134"**

UNIT III LINEAR DATA STRUCTURES – LIST 9

Abstract Data Types (ADTs) – List ADT – array-based implementation – linked list implementation –
– singly linked lists- circularly linked lists- doubly-linked lists – applications of lists –Polynomial Manipulation – All operation (Insertion, Deletion, Merge, Traversal)

1. Define ADT.

An abstract data type (ADT) is a set of operations. Classes encapsulate all the essential properties of the objects that are to be created. Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

2. How do you initialize arrays in C? Justify an array as ADT.

int a[int size];

An array is probably the most versatile or fundamental Abstract Data Type, left until now simply to show it was reasonable to consider others. An array is a finite sequence of storage cells, for which the following operations are defined:

create(A,N) creates an array A with storage for N items;

A[i]=item stores item in the i^{th} position in the array A; and

A[i] returns the value of the item stored in the i^{th} position in the array A.

3. How is an array of structures initialized? Give example.

struct student

{

intRegNo;

charSName[15];

}s[5];

4. What is the importance of header nodes?

The header node is fixed throughout the iteration and its next field will always hold a reference to the first element of the list, whatever removals take place. Linked lists use the dummy header node as a permanent feature, in order to deal with situations like this.

5.How do you implement an Adjacency List?

A common way to implement an Adjacency List is an array of [1..n] lists. Each array index has its own set of elements.

6. What are the basic operations performed on an array? Give examples.

Store & retrieve elements. create(A,N) creates an array A with storage for N items;

A[i]=item stores item in the i^{th} position in the array A; and

A[i] returns the value of the item stored in the i^{th} position in the array A.

7. What is a circular linked list? Give example.

Circular linked list is a linked list in which the last node of the list points to the first node in the list. A good example of an application where circular linked list should be used is a timesharing problem solved by the operating system. In a timesharing environment, the operating system must maintain a list of present users and must alternately allow each user to use a small slice of CPU time, one user at a time. The operating system will pick a user; let him/her use a small amount of CPU time and then move on to the next user, etc. For this application, there should be no NIL pointers unless there is absolutely no one requesting CPU time.

8. Write simple code to traverse through DLL.

Forwards

node := list.firstNode

while node ≠ null

<do something with node.data>

node := node.next

Backwards

node := list.lastNode

while node ≠ null

<do something with node.data>

node := node.prev

9. What is data structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

10. List out the areas in which data structures are applied extensively?

The names of areas are:

Compiler Design, Operating System, Database Management System, Statistical analysis package, Numerical Analysis, Graphics, Artificial Intelligence, Simulation

11. What are the major data structures used in the following areas : RDBMS, Network data model & Hierarchical data model.

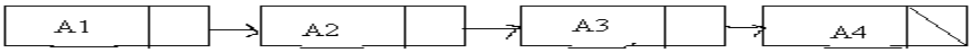
The major data structures used are as follows: RDBMS - Array (i.e. Array of structures), Network data model - Graph, Hierarchical data model - Trees

12. Mention the applications of list.

1.Polynomial ADT 2. Radix Sort 3. Multilist

13.**Define Linked Lists**

Linked list consists of a series of structures, which are not necessarily adjacent inmemory. Each structure contains the element and a pointer to a structure containing itssuccessor. We call this theNext Pointer. The last cell'sNext pointer points to NULL.



14. **State the different types of linked lists**

The different types of linked list include singly linked list, doubly linked list and

circular linked list.

15. **List out the advantages of using a linked list**

- It is not necessary to specify the number of elements in a linked list during its

  declaration

- Linked list can grow and shrink in size depending upon the insertion and deletion that occurs in the list

- Insertions and deletions at any place in a list can be handled easily and efficiently

- A linked list does not waste any memory space

16. List out the disadvantages of using a linked list

- Searching a particular element in a list is difficult and time consuming
- A linked list will use more storage space than an array to store the same number

of elements

17.**State the difference between arrays and linked lists**

Arrays	Linked Lists
Size of an array is fixed	Size of a list is variable
It is necessary to specify the numberof elements duringdeclaration.	It is not necessary to specify thenumber of elements duringdeclaration
Insertions and deletions are	Insertions and deletions are carried
It occupies less memory than alinked list for the same number ofelements	It occupies more memory

18.**List the basic operations carried out in a linked list**

The basic operations carried out in a linked list include:

- Creation of a list
- Insertion of a node
- Deletion of a node
- Find previous and successor node
- Traversal of the list

19.**What are advantages of doubly linked list over singly linked list?**

In a singly linked list, You can traverse (move) in a singly linked list in only one direction (i.e. from head to null in a list). You can't traverse the list in the reverse direction (i.e. ., from null to head in a list.)For some applications, especially those where it is necessary to traverse list in both direction, doubly linked list work much better than singly linked list. Doubly linked list is an advanced form of a singly linked list, in which each node contains three fields namely,Previous address field.Datafield.Next address field.

20.**What are the Advantagesof Modularity?**

1. Modules can be compiled separately which makes debugging process easier.
2. Several modules can be implemented and executed simultaneously.
3. Modules can be easily enhanced.

21.Write the routine to check whether the single linked list is empty or not.

```
intIsEmpty (List L) /*Returns 1 if L is empty */
{
if (L ->    Next = = NULL)
return (1);
```

```
}
```

22.Write the find previous routine in singly linked list

```
positionFindPrevious (int X, List L)
{
/* Returns the position of the predecessor */
position P;
P = L;
while (P ->    Next ! = Null && P ->Next     Element ! = X)
P = P ->Next;

return P;
}
```

23.**Write the find next routine in singly linked list**

```
positionFindNext (int X, List L)
{
/*Returns the position of its successor */
P = L ->Next;

while (P    Next! = NULL && P    Element ! = X)
P = P→Next;
returnP→Next;
}
```

24.What is radix sort?

Radix Sort is the generalised form of Bucket sort. It can be performed using buckets from 0 to 9.

In FirstPass, all the elements are sorted according to the least significant bit.

In second pass, the numbers are arranged according to the next least significant bit and so on this process is repeated until it reaches the most significant bits of all numbers.

The numbers of passes in a Radix Sort depends upon the number of digits in the numbers given.

25.What is doubly linked list?

A Doubly linked list is a linked list in which each node has three fields namely data field, forward link (FLINK) and Backward Link (BLINK). FLINK points to the successor node in the list whereas BLINK points to the predecessor node.

PART-B

1. Explain about single linked list with the algorithm and example.
2. Explain about doubly linked list.
3. a)Write an algorithm and implementation to count number of nodes in the linear linked list.
b)How two sorted linked lists are merged to produce a single sorted list? Give an algorithm and explain them with example.
4.Given two lists L1 and L2, Write the routine to compute L1 ∩ L2 using basic operations. **(Hint: for efficient performance, sort the lists).**
5.Explain about circular linked list
6.Explain about the applications of list.
7.Explain about polynomial addition.
8.Explain about polynomial subtraction.
9.Explain all the types of linked list.
10 a)Write the routine for searching element and find the successor node in a singly linked list.(10)
b)Write the advantages and disadvantages of linked list.(6)

UNIT IV LINEAR DATA STRUCTURES – STACKS, QUEUES
9

Stack ADT – Evaluating arithmetic expressions- other applications- Queue ADT – circular queue implementation – Double ended Queues – applications of queues

1. Define a stack

Stack is an ordered collection of elements in which insertions and deletions are restricted to one end. The end from which elements are added and/or removed is referred to as top of the stack. Stacks are also referred as piles, push-down lists and last-in-first-out (LIFO) lists.

2. List out the basic operations that can be performed on a stack

The basic operations that can be performed on a stack are

- Push operation
- Pop operation
- Peek operation
- Empty check
- Fully occupied check

3. State the different ways of representing expressions

The different ways of representing expressions are

- Infix Notation
- Prefix Notation
- Postfix Notation

4. State the advantages of using infix notations

- It is the mathematical way of representing the expression
- It is easier to see visually which operation is done from first to last

5. State the advantages of using postfix notations

- Need not worry about the rules of precedence
- Need not worry about the rules for right to left associativity
- Need not need parenthesis to override the above rules

6. State the rules to be followed during infix to postfix conversions

- Fully parenthesize the expression starting from left to right. During parenthesizing, the operators having higher precedence are first parenthesized
- Move the operators one by one to their right, such that each operator replaces their corresponding right parenthesis
- The part of the expression, which has been converted into postfix is to be treated as single operand

7. State the rules to be followed during infix to prefix conversions

- Fully parenthesize the expression starting from left to right. During parenthesizing, the operators having higher precedence are first parenthesized
- Move the operators one by one to their left, such that each operator replaces their corresponding left parenthesis
- The part of the expression, which has been converted into prefix is to be treated as single operand
- Once the expression is converted into prefix form, remove all parenthesis

8. State the difference between stacks and linked lists

The difference between stacks and linked lists is that insertions and deletions may occur anywhere in a linked list, but only at the top of the stack

9. Mention the advantages of representing stacks using linked lists than arrays

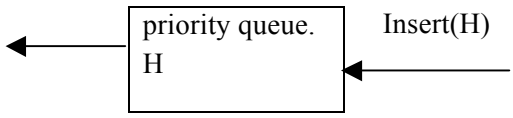
- It is not necessary to specify the number of elements to be stored in a stack during its declaration, since memory is allocated dynamically at run time when an element is added to the stack
- Insertions and deletions can be handled easily and efficiently
- Linked list representation of stacks can grow and shrink in size without wasting memory space,

depending upon the insertion and deletion that occurs in the list
• Multiple stacks can be represented efficiently using a chain for each stack

10. What does a Priority Queue mean?

Queue is a kind of data structure, which stores and retries the data as First In First Out. But priority queue disturbs the flow of FIFO and acts as per the priority of the job in the operating system

Del-min(H)



11. State the difference between queues and linked lists

The difference between queues and linked lists is that insertions and deletions may occur anywhere in the linked list, but in queues insertions can be made only in the rear end and deletions can be made only in the front end.

12. Define a Deque

Deque (Double-Ended Queue) is another form of a queue in which insertions and deletions are made at both the front and rear ends of the queue. There are two variations of a deque, namely, input restricted deque and output restricted deque. The input restricted deque allows insertion at one end (it can be either front or rear) only. The output restricted deque allows deletion at one end (it can be either front or rear) only.

13. State the difference between persistent and ephemeral data structure

Persistent data structures are the data structures which retain their previous state and modifications can be done by performing certain operations on it. Eg) Stack Ephemeral data structures are the data structures which cannot retain its previous state. Eg) Queues

14. What are the types of queues?

• Linear Queues – The queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a linear queue in only one direction ie) from front to rear.
• Circular Queues – Another form of linear queue in which the last position is connected to the first position of the list. The circular queue is similar to linear queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a circular queue in only one direction ie) from front to rear.
• Double-Ended-Queue – Another form of queue in which insertions and deletions are made at both the front and rear ends of the queue.

15. List the applications of stacks

• Towers of Hanoi
• Reversing a string
• Balanced parenthesis
• Recursion using stack
• Evaluation of arithmetic expressions

16. List the applications of queues

• Jobs submitted to printer
• Real life line
• Calls to large companies
• Access to limited resources in Universities
• Accessing files from file server.

17. What are the postfix and prefix forms of the expression?

A+B*(C-D)/(P-R)
Postfix form: ABCD-*PR-/+
Prefix form: +A/*B-CD-PR

18. Explain the usage of stack in recursive algorithm implementation?

In recursive algorithms, stack data structures is used to store the return address when a recursive call is encountered and also to store the values of all the parameters essential to the current state of the procedure.

19. Write down the operations that can be done with queue data structure?

Queue is a first - in -first out list. The operations that can be done with queue are addition and deletion.

20.State Tower of Honoi.

The objective is to transfer the entire disk to tower 1 to entire tower 3 using tower2. The rules to be followed in moving the disk from tower1 to tower 3 using tower 2 are as follows,
Only one disc can be moved at a time
Only the top disc on any tower can be moved to any other tower.
A larger disc cannot be placed on a smaller disc.

PART B

1.Explain in detail about array stack and linked stack.
2.Explain in detail about array queue and linked queue.
3.Explain the applications of stack with suitable example.
4.Explain about Circular queue with example.
5.Explain about Double ended Queues .
6.a)Convert the infix expression (a*b)+((c*g)-(e/f)) into reverse polish notation.
b) Write an algorithm for convert infix expression to postfix expression with an example of (A+(B*C-(D/E^F)*G)*H).
7.Explain the implementation of Evaluating Postfix Expression.
8.Explain the implementation of Converting Infix to Postfix Expression.
9.a)Write the algorithm and Convert the infix expression a + b * c + (d * e + f) * g to its equivalent postfix expression.
10.a)Evaluate the postfix expression abcd+e*+f+* using stack.
b)Convert A * (B + C) * D to postfix notation.

UNIT V SORTING, SEARCHING AND HASH TECHNIQUES

Sorting algorithms: Insertion sort - Selection sort - Shell sort - Bubble sort - Quick sort –

Merge sort -Radix sort – Searching: Linear search –Binary Search Hashing: Hash Functions – Separate Chaining – Open Addressing – Rehashing – Extendible Hashing.

PART A

1. What is meant by sorting?

Ordering the data in an increasing or decreasing fashion according to some relationship among the data item is called sorting.

2. What are the two main classifications of sorting based on the source of data?

a. Internal sorting
b. External sorting

3. What is meant by external sorting?(april/may 2010)

External sorting is a process of sorting in which large blocks of data stored in storage Devices are moved to the main memory and then sorted.

4. What is meant by internal sorting?

Internal sorting is a process of sorting the data in the main memory.

5. What are the various factors to be considered in deciding a sorting algorithm?

a. Programming time

b. Execution time of the program

c. Memory needed for program environment

6. What is the main idea in Bubble sort?

The basic idea underlying the bubble sort is to pass through the file sequentially Several times. Each pass consists of comparing each element in the file with its successor (x[i] and x[i+1] and interchanging the two elements if they are not in proper order.

7. What is the basic idea of shell sort?(nov/dec 2010)

Instead of sorting the entire array at once, it is first divide the array into smaller segments, which are then separately sorted using the insertion sort.

8. What is the purpose of quick sort?

The purpose of the quick sort is to move a data item in the correct direction, just enough for to reach its final place in the array.

9. What is the advantage of quick sort?(nov/dec 2010)

Quick sort reduces unnecessary swaps and moves an item to a greater distance, in one move.

10. What is the average efficiency of heap sort?

The average efficiency of heap sort is 0 (n(log2 n)) where, n is the number of elements sorted.

11. How many passes are required for the elements to be sorted in insertion sort?

One of the simplest sorting algorithms is the insertion sort. Insertion sort consist of N-1 passes. For pass P=1 through N-1 , insertion sort ensures that the elements in positions 0 through P-1 are in sorted order .It makes use of the fact that elements in position 0 through P-1 are already known to be in sorted order .

12. Write the function in C for insertion sort ?

```
Void insertionsort(elementtype A[ ] , int N)
{
int j, p;
elementtype tmp;
for(p=1 ; p <N ;p++ )
{
tmp = a[ p] ;
for ( j=p ; j>0 && a [ j -1 ] >tmp ;j--)
a [ j ]=a [j-1 ] ;
a [ j ] = tmp ;
}
}
```

13. Write the function in c for shell sort?

```
Void Shellsort(Elementtype A[ ],int N)
{
int i , j , increment ;
elementtype tmp ;
for(elementtype=N / 2;increment > 0;increment / = 2)
For( i= increment ; i <N ; i ++)
{
tmp=A[ ];
for( j=I; j>=increment; j - =increment)
if(tmp< A[ ]=A[j – increment];
A[ j ]=A[ j – increment];
Else
Break;
A[ j ]=tmp;
}
}
```

14.. What is maxheap?
If we want the elements in the more typical increasing sorted order, we can change the ordering property so that the parent has a larger key than the child. it is called max heap.

15. Differentiate between merge sort and quick sort?
Mergesort Quicksort
1. Divide and conquer strategy Divide and conquer strategy
2. Partition by position Partition by value

16. Mention some methods for choosing the pivot element in quicksort?
1. Choosing first element
2. Generate random number
3. Median of three

17. What are the three cases that arise during the left to right scan in quicksort?
1. I and j cross each other
2. I and j do not cross each other
3. I and j points the same position

18. Define two way merge?
It is a basic external sorting in which there are two inputs and two outputs tapes.

19. Define multi way merge?
If we have extra tapes then we can expect to reduce the number of passes required to sort our input. We do this by extending two way merge to a k-way merge.

20. Define polyphase merge?
The k-way merging strategy requires the use of 2 k tapes. This could be prohibitive for some applications. It is possible to get by with only k+1 tapes.

21. What is replacement selection?
We read as many records as possible and sort them. Writing the result to some tapes. This seems like the best approach possible until one realizes that as soon as the first record is written to a output tape the memory it used becomes available for another record. If the next record on the input tape is larger than the record we have just output then it can be included in the item. Using this we can give algorithm. This is called replacement selection.

22. What are the properties involved in heapsort?
1. Structure property
2. Heap order property

23. Define articulation points.
If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.

24. What are the two stages for heap sort?
Stage 1: Construction of heap
Stage 2: Root deletion N-1 times

PART B

1. Explain Insertion sort with example.
2. Write a note on shell sort with example.
3. Discuss briefly about heap sort with example
4. Explain Merge sort with Example.
5. Explain Quick sort in detail with an eg.
6. Define Hash function. Write routines to find and insert an element in separate chaining.
7. Explain extendible hashing to resolve collision.
8. Explain open addressing with example.
9. Explain Bubble sort with example.
10. Explain Linear search with example?
11. Explain binary search with an example?
12. Explain Radix sort with an example?