Assignment-4

1) Call by value:

In call by value parameter passing method. The copy of actual parameter values are copied to formal parameters and these formal parameters are used in called function.

Ex:

```
#include<stdio.h>
void main()
{
int num1, num2;
void swap(int, int);
num1 = 10;
num2 = 20;
printf("Before swap: num1 = %d, num2 = %d", num1, num2);
swap(num1, num2);
printf("After swap: num1 = %d, num2 = %d", num1, num2);
void swap(int a, int b) {
int temp;
temp = a;
a = b;
b = temp;
}
```

Call by reference :

In call by reference parameters passing method, the memory location address of the actual parameters is copied to formal parametric. This address is used to acess the memory locations of the actual parametric is called function.

Ex :

```
#include<stdio.h>
void main()
{
int num1, num2;
void swap (int*, int*);
num1 = 20;
num2 = 20;
printf("Before swap : num1 = %d, num2 = %d", num1, num2);
swap(num1, num2);
printf("After swap : num1 = %d, num2 = %d", num1, num2);
}
void swap (int *a, int *b){
int temp;
temp = *a;
*a = *b
*b = temp;
}
```

2) #include<stdio.h>

```
void main()
{
char a[5][20];
int i,j;
printf("enter the names");
for(i=0;i<5;i++){
scanf("%s",&a[i]);
printf("sorted list of names");
for(i=65;i<122;i++)
{
for(j=0;j<5;j++)
{if(a[j][0]==i)
printf("\n%s",&[j]);
}
}
}
```

Output : Enter the names : chinni
Banni
Nani
sunny
chery

Sorted list of names :

Banni
chery
chinni
Nani
sunny

3)
```
#include <stdio.h>
int fibonacci (int num) {
if (num == 0)
{
return 0;
}
else if (num == 1){
return 1;
}
else
{
return fibonacci (num -1) + fibonacci (num - 2));
}
}
int main ()
{
int num;
printf ("enter the numbers");
scanf ("%d", &num);
for (int i = 0; i <num; i++)
{
printf ("%d", fibonacci (i));
}
return 0;
}
```

4) String handling functions:

C provides a set of standard library functions for handling strings, which are defined in the string.h header file. Some of the commonly used string handling functions in c include.

1) strcpy():

This function is used to copy find the length of given string

2) strlen():
This function is used to copy one string to another.

3) strcat():
This function is used to concatenate two strings.

4) strcmp():
This function is used to compare two strings. It returns 0 if the strings are equal, a negative value if the first string is lexicographically less than second string, and a positive value if the first string is lexicographically greater than the second string.

5) strchr():
This function is used to search for the first accurance of a given character in a string.

6) strstr():
This function is used to search for first occurance of a given substring in a string.

There are several other string handling functions in c, such as strcpy(), strcat(), strcmp() etc. These functions work similarly to the functions mentioned above, but they accept an additional arguement specifying the maximum no. of charecters to be used.

5) program to sort the given set of strings

```
#include <stdio.h>
#include <string.h>
# define Max - STRINGS 10
# define Max - LENGTH 50
void sortstrings (char strings [][Max length], int n)
{
char temp [MAX LENGTH];
for (int i=0; i<n-1; i++)
{
for (int j=0; j<n; j++)
{
if (strcmp (strings (i), strings (j))>0)
{
strcpy (temp, string [i]);
strcpy (strings [i], strings [j]);
strcpy (string [j], temp);
}
}
}
int main ()
{
char string [Max - STRIGS] [MAX - LENGTH];
int n;
printf ("Enter no. of strings");
scanf ("%d", &n);
printf ("Enter %d strings \n", n);
for (int i=0; i<n; i++)
```

```
{
scanf ("%s", strings[i]);
}
sortstrings(strings, n);
printf ("Sorted strings:\n");
for (int i=0; i<n; i++)
{
printf ("%s\n", strings[i]);
}
return 0;
}
```

(c) In, Programing a function is a block of code that performs a specific task. The structure of a user defined function in c language typically includes the following elements:

1. The function decloration, which includes the return type, function name, and list of parameters (if any) enclosed in parameter.
2. The function body, which contains the statement that are executed when the function is called

For example, the following is a simple function in c

```
int add(int a, int b)
{
int c = a+b;
return c;
}
```

Arguements: In the above example, the variables a
are the arguements passed to the function They are used to
pass data into the function.

When the function is called, The values passed as arguements
are used to perform the operations defined in the function
an the return value is to pass the results back to the
calling code.

⇒)
```
#include <stdio.h>
main()
{
int a[10], n, i;
float avg, sum=0;
printf("Enter array size");
scanf("%d", &n);
printf("Enter array elements");
for(i=0; i<n; i++){
scanf("%d", &a[i]);
 sum * =a[i]
}
print("sun=%d", sum);
avg = (float)sum/n;
Printf("Average = %f", avg);
}
```

8) Self referential structures:

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

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

-> The point to consider is that the pointer should be initialized properly before accesing, as by default it contains garbage value.

-> It plays a very important rde in creating other data structures.

-> It reduces the complexity of the program. By using this, we can easily implement these data structures efficiently.

Nested Structure:

Nested structure is a structure within the structure. One structure can be declared inside another structure can be declared inside another structure in the same way structure. members are declared inside a structure. It can be used to represent a wide variety of data.

-> These are often in using programing to store data such as the pointers on a 2D grid.

→ These are often used to represent hierarchical data, such as the content of the file system.

→ Nested structure can be used to create complex data types that are easy to understand and use.

9)

Dynamic memory allocation enables the programer to allocate memory at runtime.

→ Malloc ():

malloc stands for memory allocation. The function reverse a block of memory of the specified number of bytes.

Syntax:

ptr = (cast-type*) malloc (byte size)

→ Calloc ()

calloc stands for contiguous allocation method in C is used to dynamically allocate the specified no. of blocks of memory of the specified type.

syntax:

ptr = (cast-type*) calloc (n, element-size);

If space is insufficient allocation fill and returns a null pointer.

→ realloc ()

realloc stands for re-allocation. If the dynamically allocated memory is insufficient or more than required,

you can change the size of previous allocated memory using this function.

syntax:

```
ptr = realloc (ptr, x);
```

```
for (i=5 ; i<n ; ++i ) {
ptr [i] = i+1;
}.
printf ("elements of array are");
for (i=0 ; i<n ; ++i) {
printf ("%d", ptr [i]);
}
free(ptr);
}
return 0;
}.
```

Output:

enter no. of elements : 5
Memory is allocated
elements of array are : 1, 2, 3, 4, 5
enter new size of array : 10
memory is reallocated
The elements of array are : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

10) storage classes:

These are used to describe the features of a variable. These features basically include the scope, visibility and life-time which helps us to trace the existance of a particular variable during the runtime of a program.

→ Auto:

This is the default storage for all the variables declared inside a function or a work. Hence, the keyword auto is rarely used while writing programs. Auto variables can be only accessed and not outside them.

→ Extern:

This class simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is di assigned to it in a different block and this can be changed in a different block as well.

→ Static:

It is used to declare static variable which are popularly used while writing programs in C. These have the property of preserving their value even they are out of scope.

→ Register:

This class declares register variables that have the same functionality as that of the auto variables. The only difference is that the compiler tries to store this variables if a free registration is available.

11)
```
#include <stdio.h>
#include <string.h>
#define Max-Books 10.
struct book
{int access-no ;
char author [50];
char title [100];
int year;
float price;
}
int main ()
{
struct book library [MAX-BOOKS];
int i, n;
printf ("Enter no. of Books");
scanf ("%d", &n);
for (i=0; i<n; i++)
{
printf ("Enter details for books %d \n", i+1);
printf ("Access Number:");
scanf ("%d", & library (i). access-no);
printf ("Author");
scanf ("%s", library [i], author);
printf ("title:");
scanf ("%s", library (i); title);
printf ("year of publication:");
scanf ("%d", & library (i). price);
}
printf ("\n library catalogue. \n");
```

```
for (i=0; i<n; i++)
{
printf("Enter details for book %d \n", i+1);
printf("Access Number : %d \n", library[i], access_no);
Printf ("Author : %.s \n", library[i], title);
printf ("Title : %.s \n", library[i], title);
printf ("price : %.2f \n", library[i] price);
}
return 0;
}
```

12). Command line arguement :

These are given after the name of the program in command - line shell of operating systems. Command line arguements are passed to the main() method.

Syntax :

```
int main (int argc, char *argv[])
```

argc counts the number of arguements on the comand lie and argv[] is a pointer array, which holds pointers to the type char which points to the arguements.

Ex:

```
#include <stdio.h>
int i;
if (argc >=2)
{
printf (" The arguements are : ");
for (i=1; i< argc ; i++) {
printf ("%s\t" , argv[i]);
}
}
else
{
printf("argument list is empty");
}
return 0 ;
}
```

Output :

Arguement List is empty .

13) Pointer :

It is a variable that stores the memory adress of another as its value pointer is created with the * operator .

There are few operations that are allowed to perform on pointers .

* Increment | Decrement :

Increment : When pointer is incremented , it actually

increments by the number equals to the & then added to pointer.

Ex:

```
#include <stdio.h>
int main () {
int n = 4;
int ptr1, ptr2;
ptr1 = &n;
ptr2 = &n;
ptr2 = ptr2 + 3;
printf ("Pointer Ptr2: %p", ptr2);
}
```

Output : Pointer ptr2 = 0x7ffca3f3da8

Subtraction :

```
#include <stdio.h>
int main () {
int n = 4;
int *ptr1, *ptr2;
ptr1 = &n;
ptr2 = &n;
ptr2 = ptr2 - 3;
printf ("Pointer ptr2 %p", ptr1);
return 0;
}
```

14) File:

It is collection of data stored in the secondary memory. it is used to storing informations that can be passed by programs File mode is categorized into 4 type of modes.

→ Create mode:

Opens the specified file and positions is to the begining To create a new file, open the file in create mode user can't read, position a file opened with create mode

→ Read mode:

This mode opens a file for the reading of data. Read mode opens a file to the begining.

→ Update mode:

This mode allows both reading & writing of data. Update mode life opens a file to the begining

→ Append mode:

This allows writing data to the end of a file. User can't read position or rewind a file opened with append mode.

15)

```
#include <stdio.h>
int main()
{
FILE *fptr1, *fptr2;
```

```
char filename[100], c;
printf("enter file name to open:\n");
scanf("%s", filename);
fptr1 = fopen(filename, "r");
if (fptr1 == NULL) {
printf("can't open file %s\n", filename);
exit(0);
}
printf("enter file name to open for");
scanf("%s", filename);
fptr2 = fopen(filename, "w");
if (fptr2 == NULL) {
printf("can't open file %s\n", filename);
exit(0);
}
c = fgetc(fptr1);
while (c != EOF) {
fputc(c, fptr2);
c = fgetc(fptr1);
}
printf("\n contents copied to %s", filename);
fclose(fptr1);
fclose(fptr2);
return 0;
}
```

18) F scanf(); This function is used to read formatted input from a file. It works similarly to the scanf() function but it takes an addition file pointer as the first arguement for ex, the following code reads an integer, a string and a float from a file called "data.txt"

```
FILE *fp;
int i;
char str[100];
FP = f open ("data txt.", r);
     F scanf (ep, %d %s %f", &; str, &r);
     printf (Read ; %d %s %f", i, str, f);
     f close (FP);
```

F printf(); This function is used to react a line of text from a file. It takes a file pointer, a buffer to store the read. text, and the Maximum no. of charecters to read as arguements. for

Ex: The following code reads a line of text from a file

```
FILE * fp;
int i=42;
char str[] = "Helloworld";
float f = 3.14;
FP = fopen (data.txt, 'w");
f printf (FP, %d %s %f;", i, str, f);
F close (FP);
```