CSE

Assignment-4

K.N.S. Kusumapriya.
AP22110011048
SEC:- P.

1. Explain about call by value and call by reference with suitable examples.

Ans:- Call by value:- In this method, the function receives a copy of the argument's value. This means that any changes made to the argument within the function have no effect on the original value outside of the function.

Example:-

```
void increment (int x)
{
  x++;
}
int main()
{
  int a=5;
  increment (a);
  Printf("%d",a);
}
```

output: 5

Call by reference:- Here, the function receives a pointer to the argument. This means that any changes made to argument within the function will affect the original value outside of the function.

Example:-

```
void increment(int* x)
{
  (*x)++;
}
```

```
int main ()
{
  int a=5;
  increment (&a);
  Printf("%d", a);
  }
```

output:

6.

2. Write a C program for multiplication of 2 Matrices

```
#include<stdio.h>
 int main ()
{
  int a[10][10], b[10][10], c[10][10], i, j, k, m, n, p, q;
  Printf("Enter no. of rows & Columns of Matrix A:");
  scanf ("%d %d", &m, &n);
  Printf ("Enter elements of Matrix A:");
   for (i=0; i<m; i++)
   {
     for (j=0 ; j<n; j++)
     {
        scanf("%d", &a[i][j]);
      }
     }
  Printf ("Enter no. of rows & columns of Matrix B:");
  scanf ("%d %d", &p, &q);
  Printf ("Enter elements of Matrix B:");
   for (i=0; i<p; i++)
    {
       for (j=0; j<q, j++)
     {
       scanf("%d", &b[i][j]);
      }
    }
```

```
if(n!= p)
{
Printf("No.of columns in Matrix A must be equal
        to No.of rows in Matrix B\n");

  return 0;

}
for(i=0; i<m; i++)
 {
   for(j=0; j<q ; j++)
    {
       c[i][j]=0;
        for(k=0; k<n; k++) {

          c[i][j] += a[i][k]* b[k][j]
        }
      }
     }
    Printf("Product of given 2 Matrices:\n");
    for(i=0; i<m; i++)
    {
      for(j=0; j<q; j++)
      {
        Printf("%d", c[i][j]);
      }
        Printf("\n");
      }
      return 0;
    }
```

3. Write a C program to implement Fibonacci Series using recursion.

```
#include <stdio.h>
int fibonacci (int n)
{
  if (n <= 0)
  return 0;
  if (n == 1)
  return 1;
  else
  return fibonacci (n-1) + fibonacci (n-2);
}
int main ()
{
  int n, i;
  Printf ("Enter no. of terms : ");
  Scanf ("%d", &n);
  Printf ("Fibonacci Series : ");
  for (i=0; i<n ; i++)
  Printf ("%d", fibonacci(i));
  return 0;
}
```

Q: Explain about string handling functions.

A: Some of the commonly used string handling functions in C include:

→ strlen(): This function is used to find the length of a given string.

→ strcpy(): This function is used to ~~print~~ copy one string to another.

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

→ strcmp(): This function is used to compare 2 strings. It returns the value 0 if 2 strings are equal.

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

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

There are several other string handling functions in C, such as strncpy(), strncat() etc. These functions work similarly to the function mentioned above, but the accept an additional argument specifying the maximum no. of characters to be used.

5. Write a C 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 (string [i], strings[j])>0)
      {
        strcpy (temp, strings[i]);
        strcpy (strings[i], strings [j]);
        Strcpy [strings [j], temp);
  }}}}
int main ( )
{
  char strings [MAX_STRINGS] [MAX_LENGTH];
  int n;
  Printf(" Enter the no.of strings:");
  Scanf(" %d", &n);
  Printf(" Enter %d strings.\n", n);
  for (int i=0; i<n; i++)
```

```
{
 scanf("%s", strings[n]);
}
 sort strings (strings, n);
  Printf ("sorted strings: \n");
   for (int i=0; i<n ; i++)
  {
   Print f ("%s \n", strings[i]);
  }
   return 0;
  }
```

6. What do you mean by a function? Give the structures of user defined function and explain about the arguments & return values.

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

1. The function declaration, which includes return type, function name, and the list of parameters enclosed in parameters.

2. The funct^n body, which contains the statements that are executed when the function is called.

For example :-

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

Arguments:- In the above example, the variables 'a' & 'b' are the arguments passed to the function. They are used to pass data into the function.

Return values:- In the given example, the variable 'c' is the return value of the function. It is used to return a value back to the calling code. The return statement is used to return the value of the variable 'c' to the calling code.

When the function is called, the values passed as arguments are used to perform the operations defined in the function, and the return value is to pass the results back to calling code

7. Write a program to read, calculate average and print student marks using array of structure.

```
#include <stdio.h>
  struct student.
{
    int roll_no;
    char name[20];
    float marks[3];
    float average;
};
int main()
{
    int i,j,n;
    struct students s[10];
    Printf("Enter the no. of students:");
    scanf("%d", &n);
```

```
for(i=0;i<n;i++)
{
Printf("Enter details for students %d:\n",i++);
Print("Roll number:");
scanf("%d",&s[i].roll_no);
 Printf("Name:");
scanf("%s",s[i].name);
for(j=0; j<3; j++)
{
 Printf("Marks is subject %d:",j+1);
 scanf("%f",&s[i].marks[j]);
 }
}
for(i=0; i<n;i++)
{
 float Sum=0;
 for(j=0; j<3; j++)
 {
 Sum+=s[i].marks[j];
 }
 s[i].average=Sum/3;
 }
 Printf("Roll Number:%d\n",s[i].roll_no);
 Printf("Name:%d\n",s[i].name);
 PrintfC"Average marks:%2f\n",s[i].average);
 }
 return 0;
 }
```

8. Differentiate between self-referential structure and nested structure with example.

A:- In C programming, a self-referential structure is a structure that contains a pointer to an instance of the same structure type.

Ex:-

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

In this example the 'node' structure contains an integer "data" and a pointer "next" to another instance of the "node" structure. This allows us to create a linked data structures, such as linked lists and trees. where each node points to the next node in the list.

→ On the other hand, a nested structure is a structure that contains another structure as a member. It is used to group related data together and to create more complex data structures

Ex:-

```
struct address {
  char street[20];
  char city[20];
  char state[20];
```

```
};
struct employee {
 int id;
 char name [20];
 struct adress addr;
};
```

In this example the "address" structure contains three character arrays for this street, city and state and the "employee" structure contains an integer 'id', a character array 'name' and a nested address structure 'addr'. This allows us to group the address details.

9. Explain three dynamic memory allocation functions with suitable examples.

In C programming, dynamic memory allocation refers to the procem of allocating memory at runtime, as opposed to compile time. There are several funct"s available in the C standard library for allocating dynamic memory, including.

1. malloc (): This function is used to allocate a block of memory ~~int~~ of a specified size. The pointer returned by malloc () points to the first byte of the allocated memory block, otherwise it returns a null pointer.

Ex:-

```
#include<stdio.h>
#include<stdlib.h>
  int main()
{
    int n,i;
    int *p;
    Printf("Enter No of elements:");
    scanf("%d",&n);
    p=(int*) malloc (n* size of (int));
    if (p==NULL)
    {
     Print (" Memory allocation failed \n");
     return 1;
    }
    for (i=0; i<n ;i++)
    {
      Printf(" Enter element %d: ",i+1);
      Scanf(" %d", &p[i]);
    }
    Printf (" Entered elements are :");
     for (i=0; i<n ;i++)
    {
      Printf (" %d", p[i]);
     }
      Printf (" \n");
       free (p);
        return 0;
     }
```

2. calloc() : This funct^n is used to allocate a block of memory for an array of a specified number of elements each of a specified size. It returns a pointer to the 1st byte of allocated memory block. Points to the first byte to the allocated memory block, otherwise it returns a null pointer.

Ex:-

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int n,i;
  int *p;
  Printf("Enter no.of elements:");
  Scanf("%d",&n);
  P=(int*) calloc(n, size of (int));
   if (P==NULL)
   {
     Printf("Memory allocation failed \n");
     return 1;
    }
   for(i=0;i<n;i++)
   {
     Printf("Enter element %d:",i+1);
      Scanf("%d",&P[i]);
    }
    Printf("Entered elements are:");
      for(i=0;i<n;i++)
    {
     Printf("%d",P[i]);
     }
```

```
Printf("\n");
free(p);
return 0;
}
```

10. Explain about storage classes:-

In C programming, a storage class is way to specify the duration and visibility of variable & funct^n. these are 4 storage classes in C.

1. Automatic:- These are local variables that are defined inside a funct^n. They are also called "local variables" & "automatic variables". They are automatically created when the funct^n is called & automatically destroyed when the funct^n returns. They are the default storage class for local variables, if no storage class is specified.

Ex:-
```
void func(){
int x;
x=5;
Printf("%d",x);
}
```

2. Register:- These are local variables they are stored in a register instead of memory. Using a register storage class can improve the performance of the program by reducing memory access time.

Ex:-

```
void func(){
register int x;
  x=5;
 Printf("%d",x);
}
```

3. Static: These are variables that retain their value blw function calls. They are also used to create variables that are only visible with a specific file, rather than being visible throughout the entire program. A variable defined as static inside a functn maintains its value between function calls.

Example:-

```
void fun() {
static int x=0;
  x++;
  Printf("%d",x);
}
```

4. Extern:- These are variables that are defined in one file and can be accessed in another file. They are used to share variables between different files (or) modules in a program. An external variable can be defined in one source file & used in another source file.

Example:-

```
//file 1.c
 int x;
  x=5;
// file 2.c
  extern int x;
   Printf("%d",x);
```

11. Develop a programme to create a library catalouge with the following members: accen number, authors, name, title of book, yeal of Publication and book Price using structures.

```
#include<stdio.h>
#include<string.h>
#define MAX_BOOKS 10
Struct book {
  int accen_no;
   Chas author[50];
   Chas title[100];
   int yeal;
   float price;
   };
  int main()
  {
    Struct book library[MAX_BOOKS];
    int i,n;
   Printf("Enter the no.of books:");
   Scanf("%d",&n);
   for(i=0; i<n; i++)
   {
    Printf("Enter details for book %d:\n",i+1);
    Printf("Accen Number:");
    Scanf("%d", &library[i].accen_no);
    Printf("Author:");
    Scanf("%s", library[i].author);
    Printf("title:");
    Scanf("%s", library[i].title);
    Printf("Yeal of publication:");
```

```
scanf("%d", &library[i].year);
Printf("Price:");
Scanf("%f", &library[i].Price);
}
Printf("\n library catalouge:\n");
for(i=0; i<n; i++){
Print f ("Accen numbes: %d\n", library[i].accen_no);
Print f ("Title: %s\n", library[i].title);
Printf ("Yeas of publication: %d\n", library[i].year);
Printf("Price: %.2f\n", library[i].Price);
}
return 0;
}
```

12. Explain about command line arguments with an example.

A:- Command-line argument are simple parameters that are given on the system's command line, and the values of these arguments are passed on to your program during Program execution. When a program starts execution without user interaction, command-line arguments are used to Pass values & files to it

Syntax:-

* Main function without arguments:

```
int main()
```

* Main function with arguments:

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

Ex:-

```
#include<stdio.h>
int main(int argc, char *argv[])
{
  int i;
  Printf("\n Program name : %s", argv[0]);
  if (argc < 2)
  {
    Printf("\n\n No argument passed through
                command line!");
  }
  else
  {
    Printf("\n Argument Supplied : ");
    for(i=1; i<argc; i++)
    {
      Printf("%s \t", argv[i]);
    }
  }
}
```

For out we shall pass arguments to our code.

→ without Argument.

output:- No argument passed through the command line!

→ Pass single argument:-

Argument Supplied : Hi, there!

→ Pass more than single argument:-

Argument Supplied : hey there scales.

13. what is a pointer? Explain pointer airthimetic operations with suitable examples.

A:- A pointer is a variable that stores the memory address of another variable. Pointers are useful for many tasks in C, including dynamic memory allocation, function pointers and passing arguments to functions.

Pointer airthmetic is the manipulation of pointers to Perform various operations like addition, subtraction, Increamant, drecreamant on pointers.

In C, pointer arithmetic is performed in the following way

Ptr++ : increamant of pointer.

ptr-- : decreamant of pointer.

'Ptr+n' : decreamant from prcavious pointer.

('ptr+n) : Adds to pointer.

'ptr-n' : subtracts n from the pointer.

Example:-

```
#include<stdio.h>
int main()
{
  int arr[]={1,2,3,4,5};
  int *ptr = arr;
  Printf("value of first element : %d\n",*ptr);
  ptr++;
  Print-f("value of second element : %d\n",*ptr);
  ptr = ptr+2;
  Printf("value of third element : %d\n",*ptr);
  ptr= ptr-1;
  Print f("value of fourth element : %d\n",*ptr);
  return 0;
}
```

Output :-

Value of first element : 1
Value of second element : 2
Value of third element : 4
Value of fourth element : 3.

14. What is a file ? Explain different modes of operating a file.

A:- In C, a file is a collection of data stored on a storage device, such as a hard drive or flash drive. Files can be created, modified, and deleted by the operating system and can be used to store various types of information, such as texts, images, videos and audio.

In C, the 'fopen( )' function is used to open a file and returns a pointer to a 'FILE' structure.

The 'fopen( )' function takes two parameters: the name of the file and the mode in which the file should be opened. The different modes are in C are :

-> 'r' : opens for reading
-> 'w' : opens for writing. The existing file
-> 'a' : opens for writing. The absent file
-> 'rb' : binary file for reading.
-> 'wb' : binary file for writing.
-> 'ab' : binary file for writing ; opened in append mode.

Ex:- opening a file in c;

```
#include <stdio.h>
int main()
{
  FILE *fp;
  char ch;
  fp = fopen("example.txt","r");
  if (fp == NULL)
  {
    Printf("Error opening file.\n");
    return 1;
  }
  while ((ch = fgetc(fp)) != EOF)
  {
    Printf("%c",en);
  }
  fclose(fp);
  return 0;
```

15. Write a program to demonstrate read and write operations on a file.

```
#include <stdio.h>
int main()
{
  FILE *fp; // FILE Pointer
  fp = fopen("example.txt","w"); // open file in writing mode.
  fprintf(fp,"writing to a file in C"); // write to file
  fp close (fp); // close the file
  fp = fopen("example.txt","r"); // open file in read mode
  chas ch;
  while ((ch= fgetc(fp)) != EOF)
  {
    Printf("%c", ch);
  }
  fclose(fp); // close the file
  return 0;
}
```

16. Explain about fscanf(), fgets(), fprintf() and fwrite() functions with suitable examples.

A:- 'fscanf()': This function used to read formated input from a file. It works similarly like a scanf() function, but it takes an additional file pointers as the first argument

Ex:-

```
FILE *fp;
int i;
char str[100];
float f;
fp = fopen("data.txt","r");
fscanf (fp,"%d%s%f",&i,str,&f);
Printf ("Read : %d %s %f", i, str, f);
fclose (fp);
```

'fgets()' : This funct^n is used to read a line of text from a file. It takes a file pointer, a buffer to store the read text, and the max no. of characters to read as arguments.

Ex :-

```
FILE *fp;
char line[100];
fp = fopen("data.txt","r");
fgets (line, size of line), fp);
Printf ("Read : %s", line);
fclose (fp);
```

'fprintf()': This function is used to write formatted output to a file. It works similary to the Printf(). function, but it takes an additional file pointers as the first argument. for example:-

```
FILE *fp;
int i=42;
char str[] = "Hello world";
float f = 3.14;
fp = fopen("data.txt", "w");
fprintf(fp, "%d %s %.f", i, str, f);
fclose(fp);
```

17. write a programme to copy one file contents to another.

```
#include <stdio.h>
int main()
{
FILE *source, *target; // file pointer
source = fopen("source.txt", "r");
if (source == NULL)
{
Printf("Could not open source file\n");
return 1;
}
target = fopen("target.txt", "w");
if (target == NULL)
{
Printf("Could not open target file\n");
fclose(source);
return 1;
}
```

```
char ch;
while ((ch = fget((source)) != EOF)
{
  fputc (ch, target);
}
Printf ("file copied successfully");
fclose (source);
fclose (target);
  return 0;
}
```

18. Explain different file handling functions with syntaxes & suitable examples.

Ans: (i) 'fopen': This function is used to open a file. It takes the name of the file and mode in which it should be opened as arguments

Syntax:-

```
FILE * fopen (const char * filename, const char* mode);
```

Example:-

```
FILE *fp;
fp = fopen("example.txt","r");
```

(ii) 'fclose (FILE *fp)' :- Funct^n used to close and open a file.

Syntax:-

```
int fclose (FILE *fp);
```

Ex:-

```
fclose(fp);
```

(iii) 'fgetc(FILE *fp)': This funct^n used to read a single characters from a file.

Syntax:-

```
int fgetc(FILE *fp);
```

Ex:-

```
int ch;
ch = fgetc(fp);
```

(iv) 'fputc(int c, FILE *fp)': This funct^n is used to write a single characters to a file.

Syntax:-

```
int fputc(int c, FILE *fp);
```

Example:-

```
fputc('A', fp);
```

(v) 'fread': This function is used to read binary data from a file. It takes a pointer to the buffer as argument.

Syntax:-

```
size_t fread(void *ptr, size_t size, size_t count,
                                        FILE *fp);
```

Ex:-

```
int data[100];
fread(data, sizeof(int), 100, fp);
```

(vi) 'fprintf' :- This funct^n used to write formatted output to a file. It takes a file pointer, a format string, and a variable number of arguments.

Syntax:

```
int fprintf (FILE *fp, const char * format, ...);
```

Example:-

```
FILE *fp;
int i = 18;
float f = 3.14;
char str[] = "Hello World";
fp = fopen("example.txt", "w");
fprintf (fp "Integer: %d, float: %f, string: %s",
                    i, f, str);
fclose (fp);
```