North East University Bangladesh

Dept of Computer Science and Engineering

Assignment for Semester final Summer 2020

Course Code : CSE 113

Course title : Structured Programming Language

Submitted By :

Mansura Mokbul

Id : 2001030200 28

2nd Semester

(Section - B)

# Answer to the question number : 01

a) **Recursion :** The process in which a function calls itself is called recusion and the corresponding function is called as recursive function.

A recursive function calls itself untill a base condition is true, and execution stops.

**Syntax :**

```
void recurse()
{
    --- ---
    recurse();
    --- ---
}
int main()
{
    --- ---
    recurse();
    --- ---
}
```

Example:

```
#include <stdio.h>
int sum(int n){
    if (n<=1)
        return n;
    return (n+ sum(n-1));
}
int main (){
    int n,result;
    printf("Enter a positive integer : ");
    scanf("%d", &n);
    result = sum(n);
    printf("sum = %d", result);
    return 0;
}
```

Output:

```
Enter a positive integer : 10
sum = 55
```

b) Dynamic memory allocation: Dynamic memory allocatim is manual allocation and freeing of memory according to our programming needs.

It is mannaged and served with pointers.

Library functions : <stdlib.h>

→ malloc (): peserves a block of memory of the specified number of byten.

ptr = (castType *) malloc (size);

ptr = (float *) malloc (100 * size of (float));

→ calloc (): allocates memory and initializes all bits to zero.

ptr = (float*) calloc (25, size of (float));

→ realloc (): change the size of previously allocated memory

ptr = realloc (ptr, new_size);

→ free() : used to release dynamically allocated memory.

free (ptr);

## Answer to the question number : 02

Difference between actual and formal parameter function :

Actual parameters are the values that are passed to the function when it is invoked.

On the other hand, Formal parameters are the variables defined by the function that recives values when the function is called.

## Answer to the question number : 03

Difference between call by value and call by reference:

**Call by value:** In this method values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory location.

**Example:**

```
#include <stdio.h>
long factorial (long n){
    long i, fact = 1;
    for (i=1; i<=n; i++){
    fact *= i;
    }
    return fact;
}
int main (){
    long n, result;
    printf("Enter a number : ");
    scanf ("%ld", &n);
    result = factorial (n);
    printf ("factorial of %ld is %ld", n, result);
    return 0;}
```

Output:

```
Enter a number : 10
factorial of 10 is 3628800
```

Call by reference: Both actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.

Example:

```
#include<stdio.h>
long factorial(long n){
    long i, fact=1;
    for(i=2; i<=n; i++){
    fact *= i;
    }
    return fact;
}

int main(){
    long *n, *result;
    printf("Enter a number : ");
    scanf("%ld", &n);
    *result = factorial(n);
    printf("Factorial of %ld is %ld", *n, *result);
    return 0;
}
```

## Answer to the question number : 4

Two conditions of recursive function are :

i) A recursive function must have a base case.

ii) A recursive function must call itself recursively.

## Answer to the question number : 5

Fibonacci series :

```
#include<stdio.h>
int fibo(int n);
int main(){
    int n,i;
    printf("Enter element : ");
    scanf("%d",&n);
    printf("Fibonacci series : \n");
    for(i=0; i<n; i++)
        printf("%d\t\n",fibo(i));
    getch();
    return 0;
}
int fibo(int x){
    if(x==0 || x==1)
        return x;
    else return (fibo(x-2)+fibo(x-1));
}
```

# Answer to the questin number : 06

## Use of break statement :

The break statement ends the loop immediately when it is encountered.

## Example :

```
#include<stdio.h>
int main(){
    int i;
    double number, sum=0;
    for(i=1; i<=5; i++){
        printf("Enter a n%d :", i);
        scanf("%lf", &number);
        if(number<0){
            break;
        }
        sum+=number;
    }
    printf("Sum = %.2lf", sum);
    return 0;
}
```

### Output :

```
Enter a n1 : 1
Enter a n2 : 2
Enter a n3 : 3
Enter a n4 : 4
Enter a n5 : 5
Sum = 15.00
```

### Another output from same code :

```
Enter a n1 : 5
Enter a n2 : -2
Sum = 5.00
```

Use of continue statement: The continue statement skips the current iteration of the loop and continues with the next iteration.

Example:

```
#include<stdio.h>
int main(){
    int i;
    double number, sum=0;
    for(i=2; i<=5; i++){
        printf("Enter a n%d: ", i);
        scanf("%lf", &number);
        if(number<0){
            continue;
        }
        sum+=number;
    }
    printf("Sum = %.2lf", sum);
    return 0;
}
```

Output:

```
Enter a n1: 5
Enter a n2: -2
Enter a n3: 4
Enter a n4: -1
Enter a n5: 6
Sum = 15.00
```

## Answer to the question number : 07

Array different from ordinary variable :

An array is a variable that can store multiple values (same type). But an ordinary variable hold a single value.

And array is always declared, initialized, and accessed using subscript whereas ordanary variable do not have subscript.

## Answer to the question number : 08

When we pass the address of an array while calling a function then this is called function call by reference. And when we pass an address as an argument, the function declaration should have a pointer as a parameter to receive the passed address.

In a functim if we change the array (either its main functim or any other functim where its called) then the valve of arrays are also change.

## Answer to the question number : 9

Structure : Structure is a user-defined datatype in C language which allows us to combine data of different types together.

It can store multiple values and different datatypes in together.

Syntax :

```
struct struct_name {
    DataType var1;
    DataType var2;
    DataType var3;
    ----
};
```

To define a structure, must use the struct statement. The struct statement defines a new data type, with more than one member.

Declaration a structure : We will declare the variables of struct types. Variables will be declared separately

or with the definition.

1. The keyword struct
2. The structure tag name
3. List of variable names separated by commas.
4. A terminating semicolon.

* struct tag

```
{
    member-list
} variable-list;
```

```
struct structname {
    Data type var1;
    Data type var2;
    Data type var3;
    ....
};
```

structure declaration

```
struct struct name {
    Data type member 1 name;
    Data type member 2 name;
    Data type member 3 name;
    -----
} var_name;
```

* struct student

```
{
    int roll no;
    char name[15];
    int mark;
    char branch[10];
} st1, st2;
```

## Answer to the question number : 10

Advantage of keyword typedef in structure:

typedef makes the code short and improves readability.

## Answer to the question number : 11

Pointer: A pointer is a variable that stores the address of another variable.

Declaration of pointer variable:

The data type of a pointer variable and the variable that it points must match.

General syntax:

```
datatype * pointer_name;
```

Examples:

```
int *p1;
double *p2;
char *p3;
float *p4;
```

Initialization of pointer variable: For initializatim address operator & is used to determin the address of a variable.

Example:

```
#include<stdio.h>
void main(){
    int a=10;
    int *ptr;      // pointer declaration
    ptr=&a;        // pointer initialization
}
```

## Answer to the questim number : 12

Malloc and calloc are used in C language for dynamic memory allocation. they obtain blocks of memory dynamically.

differences or distinguish between them:

malloc() takes a single argument, while calloc() takes two. and malloc() dres not initialize the memory allocated, while calloc() initializes the allocated memory to zero.

## Answer to the question no : 13

**Dangling pointer :** Any pointer pointing to a destroyed object or which does not contain a valid address is called a dangling pointer.

## Answer to the question number : 14

```
#include<stdio.h>
int main(){
    int r,c,n;
    printf("Enter a number : ");
    scanf("%d",&n);
    for(r=1;r<=n;r++){
        for(c=1;c<=n-r;c++){
            printf(" ");
        }
        for(c=1;c<=r;c++){
            printf("%d",r);
        }
        printf("\n");
    }
    return 0;
}
```

## Answer to the question number : 15

a)
```
#include<stdio.h>
int main(){
    FILE *fp;
    char ch;
    fp=fopen("E:\\string.txt", "r");
    printf("Enter string:");
    scanf("%c", &ch);
    fprintf(fp, "%c", ch);
    fclose(fp);
    return 0;
}
```

b)
```
#include <stdio.h>
int main (){
    FILE *fp1, *fp2;
    char ch;
    fp1 = fopen ("input.c", "r");
    fp2 = fopen ("output.c", "w");
    ch = fgetc (fp1);
    while (ch != EOF){
        fputc (ch, fp2);
        ch = fgetc (fp1);
    }
    fclose (fp1);
    fclose (fp2);
    printf ("copied finish");
}
```