**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **19/06/2020** | **Name:** | **Kirti B S** |
| **Course:** | **C Programming** | **USN:** | **4AL18EC026** |
| **Topic:** | * **Structures and Unions** * **Memory Management** | **Semester & Section:** | **4th sem ‘A’ section.** |
| **Github Repository:** | **Kirti BS** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **C:\Users\hello\Pictures\Screenshots\Screenshot (284).png**  **C:\Users\hello\Pictures\Screenshots\Screenshot (285).png** |
| **REPORT**  **In today’s session I have learnt about:**  **Chapter 1: Structures and Unions**   * **Structures: In C programming, a struct (or structure) is a collection of variables (can be of different types) under a single name. To define a struct, the struct keyword is used.**   **Syntax of struct:**  **struct structurename**  **{**  **datatype member 1;**  **datatype member 2;**  **};**  **Example for struct:**  **struct Person**  **{**  **char name[50];**  **Int city no;**  **float salary;**  **};**   * **Working with Structures:** * **Unions: A Union is a special datatype available in C that allows to store different datatypes in the same memory locations. We can define an Union with many members, but only one member can contain the value at any given time. Unions provide an efficient way of using the same memory location for multiple-purpose.** * **Syntax of Union:**   **union unionname**  **{**  **datatype member 1;**  **datatype member 2;**  **};**   * **Example for union:**   **union Person**  **{**  **char name[50];**  **Int city no;**  **float salary;**  **};**   * **Working with Unions:**   **Chapter 2: Memory Management**   * **Working with memory: In C, the library function malloc is used to allocate a block of memory on the heap. The program accesses this block of memory via a pointer that malloc returns. When the memory is no longer needed, the pointer is passed to free which deallocates the memory so that it can be used for other purposes.** * **The Malloc function: malloc or memory allocation method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It initializes each block with default garbage value.** * **Syntax of malloc function:**   **ptr = (cast-type\*) malloc(byte-size);**   * **Example of malloc function:**   **ptr = (int\*) malloc(100\*sizeof(int));**  **Since the size of int is 4 bytes, this statement will allocate 400 bytes memory. And the pointer ptr holds the address of the first byte in the allocated memory.**   * **Calloc and Realloc:**   **Calloc: calloc or contagious reallocation method in C is used to dynamicallu allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value ‘0’.**   * **Syntax of calloc:**   **ptr = (cast-type\*) calloc(n, element-size);**   * **Example of calloc:**   **ptr = (float\*) calloc(25, sizeof(float));**  **This statement allocates contiguous space in memory for 25 elements each with the size of the float.**  **Realloc: Realloc or re-allocation method in C is used to dynamically change the memory allocation of a previously allocated memory.**   * **Syntax of Realloc:**   **ptr = realloc(ptr, newsize);**  **where ptr is reallocated with new size ‘newsize’.**   * **Dynamic Strings and Arrays:**   **When a Dynamic string is allocated, a free element is found in the array and s and size are set. If the array needs to grow, Realloc is used to extend it automatically. Given a char\* to a dynamic string, it is possible to scan the array looking for the corresponding element.** |