

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | CSC202A |
| **Course Name** | Data structure and Algorithms |
| **Programme** | B. Tech |
| **Department** | Computer Science & Engineering |
| **Faculty** | Faculty of Engineering Technology |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | SUBHENDU MAJI |
| **Reg. No** | 18ETCS002121 |
| **Semester/Year** | 3RD / 2019 |
| **Course Leader/s** | Vaishali R Kulkarni |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name | SUBHENDU MAJI | | | | | | | |
| Reg. No | 18ETCS002121 | | | | | | | |
| Programme | B. Tech | | | | | Semester/Year | 3rd / 2019 | |
| Course Code | CSC202A | | | | | | | |
| Course Title | Data structure and Algorithms | | | | | | | |
| Course Date |  | | To | |  | | | |
| Course Leader | Vaishali R Kulkarni, Dr Pushphavathi T P, G. Roopa | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc21640155)

[**Contents** iii](#_Toc21640156)

[Marking Scheme iv](#_Toc21640157)

[**Question No. 1** 6](#_Toc21640158)

[A1.1 Static memory allocation 6](#_Toc21640159)

[A1.2 Dynamic memory allocation 6](#_Toc21640160)

[A1.3 Comparative analysis 6](#_Toc21640161)

[**Question No. 2** 7](#_Toc21640162)

[B1.1 Classful addressing in networks 7](#_Toc21640163)

[B1.2 A Data structures used 7](#_Toc21640164)

[B1.3 A Validated C Program 7](#_Toc21640165)

[**Question No. 3** 8](#_Toc21640166)

[B2.1 Plagiarism rules and threshold 8](#_Toc21640167)

[B2.2 Pseudocode for checking plagiarized content 8](#_Toc21640168)

[B2.3 Validated C Program 8](#_Toc21640169)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Faculty of Engineering and Technology** | | |
|  | **Ramaiah University of Applied Sciences** | | |
| Department | Computer Science and Engineering | Programme | B. Tech |
| Semester/Batch | 03/2019 | | |
| Course Code | CSC202A | Course Title | Data structure and Algorithms |
| Course Leader | Vaishali R Kulkarni, Dr Pushphavathi T P, G. Roopa | | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assignment - 01** | | | | | | |  |  | | | |
| Register No | | **18ETCS002121** | | | Name of Student | |  | **SUBHENDU MAJI** | | | |
| Sections |  | Marking Scheme | | | | |  | **Marks** | | | |
| Max. Marks |  | | First  Examiner  Marks | Moderator Marks |
| **Part**  **-**  **A** |  | | | | | |  |  | | | |
| A 1.1 | Static memory allocation | | | | | 01 |  | |  |  |
| A 1.2 | Dynamic memory allocation | | | | | 02 |  | |  |  |
| A 1.3 | Comparative analysis | | | | | 02 |  | |  |  |
|  | **Part-A Max Marks** | | | | | **05** |  | |  |  |
| **Part B 1** |  | | | | | |  |  | | | |
| B 1.1 **1** | Classful addressing in networks | | | | | 02 |  | |  |  |
| B 1.2 | Data structures used | | | | | 04 |  | |  |  |
| B 1.3 | ValidatedC Program | | | | | 04 |  | |  |  |
|  | **Part-B 1 Max Marks** | | | | | **10** |  | |  |  |
| **Part B 2** |  | | | | | |  |  | | | |
| B2.1 | Plagiarism rules and threshold | | | | | 02 |  | |  |  |
| B2.2 | Pseudocode for checking plagiarized content | | | | | 04 |  | |  |  |
| B2.3 | Validated C Program | | | | | 04 |  | |  |  |
|  | **Part-B 2 Max Marks** | | | | | **10** |  | |  |  |
|  | | | | | | | | | | | |
| **Course Marks Tabulation** | | | | | | | | | | | |
| **Component- CET B Assignment** | | | **First**  **Examiner** | **Remarks** | | **Second Examiner** | | | **Remarks** | | |
| A | | |  |  | |  | | |  | | |
| B.1 | | |  |  | |  | | |  | | |
| B.2 | | |  |  | |  | | |  | | |
| **Marks (Max 25)** | | |  |  | |  | | |  | | |
| Signature of First Examiner Signature of Second  Examiner | | | | | | | | | | | |

# **Question No. 1**

**Solution to Question No. 1:**

## A1.1 Static memory allocation

**Intro:**

**Static memory allocation** is an allocation technique which allocates a fixed amount of memory during compile time and the operating system internally uses a data structure known as Stack to manage this.

In Static Memory Allocation the memory for your data is allocated when the program starts. The size is fixed when the program is created. It applies to global variables, file scope variables, and variables qualified with static defined inside functions. This memory allocation is fixed and cannot be changed, i.e. increased or decreased after allocation. So, exact memory requirements must be known in advance.

#### Key features:

* Variables get allocated permanently
* Allocation is done before program execution
* It uses the data structure called stack for implementing static allocation
* Less efficient
* There is no memory reusability

### Example

* All the variables in the program below are statically allocated.

void play

{

int a;

}

int main()

{

int b;

int c[10];

return 1;

}

C

Copy

* In this type of allocation, you strictly allocate memory for your data at compile time. This is also called simple memory allocation. It is mostly used and very easy to application.

### Deletion of allocated memory

Deletion of memory allocated to a program is as important as allocation otherwise it results in memory leakage. Statically allocated memory is automatically released on the basis of scope, i.e., as soon as the scope of the variable is over, memory allocated get freed.

### Advantages of Static memory allocation

* Simplicity of usage.
* Efficient execution time.
* Need not worry about memory allocation/re-allocation/freeing of memory
* Variables remain permanently allocated.

### Disadvantages of Static memory allocation

* Main disadvantage is wastage of memory.
* Memory can't be freed when it is no longer needed.

## A1.2 Dynamic memory allocation

Intro:

C language requires the number of elements in an array to be specified at compile time. But we may not be able to do so always. Our initial judgement of size, if it is wrong, may cause failure of the program or wastage of memory space.  
The process of allocating memory at run time is known as **dynamic memory allocation**.

Although C does not inherently have this facility, there are four librar routines known as "memory managment functions" that can be used for allocating and freeing memory during program execution. These functions help us build complex application programs that use the available memory intelligently.

**Function and their task**

* **malloc** : Allocates request size of bytes and returns a pointer to the frst byte of the allocated space.
* **calloc** : Allocates space for an array of elements, initiaizes them to zero and then returns a pointer to the memory.
* **free** : Frees previously allocated space.
* **realloc**: Modifies the size of previously allocated space.

### Allocating a block of memory: malloc

A block of memory may be allocated using the function **malloc**. The **malloc** function reserves a block of memory of specified size and returns a pointer of type **void**. This means that we can assign it to any type of pointer. It takes the following form:

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

C

Copy

ptr is a pointer of type cast-type.

Example,

x = (int \*) malloc (100 \*sizeof(int));

On successful execution of this statement, a memory space equivalent to "100 times the size of an int'" bytes is reserved and the address of the first byte of the memory allocated is assigned to the pointer x of type of int.

### Allocating multiple blocks of memory: calloc

**calloc** is another memory allocation function that is normally used for requesting memory space at run time for storing derived data types such as arrrays and structures. while **malloc** allocates a single block of storage space, **calloc** allocates multiple blocks of storage, each of the same size, and then sets all bytes to zero. The general form of **calloc** is:

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

### Releasing the used space: free

Dynamically allocated memory created with either **calloc()** or **malloc()** doesn't get freed on their own. You must explicitly use **free()** to release the space.  
Syntax:

free (ptr);

C

Copy

This statement frees the space allocated in the memory pointed by ptr.

### Advantages of Dynamic memory allocation

* Data structures can grow and shrink according to the requirement.
  + We can allocate (create) additional storage whenever we need them.
  + We can de-allocate (free/delete) dynamic space whenever we are  
    done with them.
* Dynamic Allocation is done at run time.

### Disadvantages of Dynamic memory allocation

* As the memory is allocated during runtime, it requires more  
  time.
* Memory needs to be freed by the user when done. This is important as it is more likely to turn into bugs that are difficult to find.

### Altering the size of a block: realloc

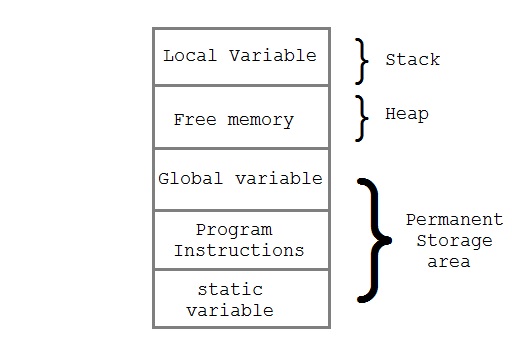
The C library function void **realloc(void \* ptr, size\_t size)** attempts to resize the memory block pointed to by ptr that was previously allocated with a call to malloc or calloc.  
Syntax:

void \*realloc(void \*ptr, size\_t size)

## A1.3 Comparative analysis

**Stack memory** is allocated during compilation time execution. This is known as static memory allocation.

Whereas, **heap memory** is allocated at run-time compilation. This is know as dynamic memory allocation.



|  |  |
| --- | --- |
| **STATIC MEMORY ALLOCATION** | **DYNAMIC MEMORY ALLOCATION** |
| Memory is allocated before the execution of the program begins (During Compilation). | Memory is allocated during the execution of the program. |
| Variables remain permanently allocated. | Allocated only when program unit is active. |
| In this type of allocation Memory cannot be resized after the initial allocation. | In this type of allocation Memory can be dynamically expanded and shrunk as necessary. |
| Implemented using stacks. | Implemented using heap. |
| Faster execution than Dynamic. | Slower execution than static. |
| It is less efficient than Dynamic allocation strategy. | It is more efficient than Static allocation strategy. |
| Implementation of this type of allocation is simple. | Implementation of this type of allocation is complicated. |
| Memory cannot be reuse when it is no longer needed. | Memory can be freed when it is no longer needed & reuse or reallocate during execution. |

# **Question No. 2**

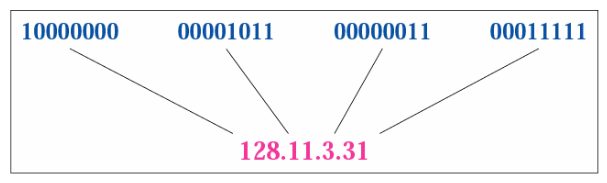
**Solution to Question No. 2:**

## B1.1 Classful addressing in networks

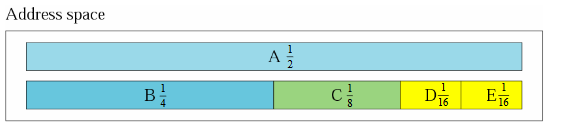
An IP address is a 32-bit address that identifies a connection to the Internet.

The IP addresses are universally unique.

The address space of IPv4 is 232or 4,294,967,296.•IP address is written as a Binary (hexadecimal) or a Dotted-Decimal (w/out leading zeros) notation.



The IP address space (all possible IP values) is divided into five classes: A, B, C, D, and E.

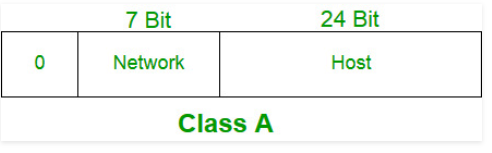


**Class A:**

IP address belonging to class A are assigned to the networks that contain a large number of hosts.

* The network ID is 8 bits long.
* The host ID is 24 bits long.

The higher order bit of the first octet in class A is always set to 0. The remaining 7 bits in first octet are used to determine network ID. The 24 bits of host ID are used to determine the host in any network. The default subnet mask for class A is 255.x.x.x.



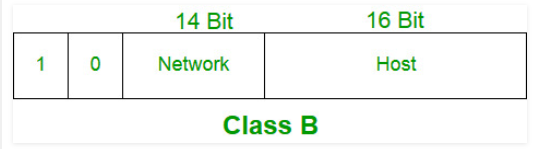
IP addresses belonging to class A ranges from 1.x.x.x – 126.x.x.x

**Class B:**

IP address belonging to class B are assigned to the networks that ranges from medium-sized to large-sized networks.

* The network ID is 16 bits long.
* The host ID is 16 bits long.

The higher order bits of the first octet of IP addresses of class B are always set to 10. The remaining 14 bits are used to determine network ID. The 16 bits of host ID is used to determine the host in any network. The default sub-net mask for class B is 255.255.x.x.



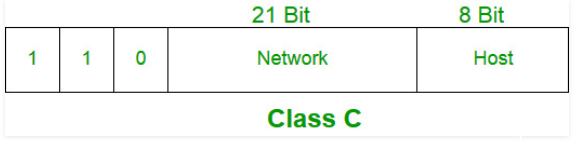
IP addresses belonging to class B ranges from 128.0.x.x – 191.255.x.x.

**Class C:**

IP address belonging to class C are assigned to small-sized networks.

* The network ID is 24 bits long.
* The host ID is 8 bits long.

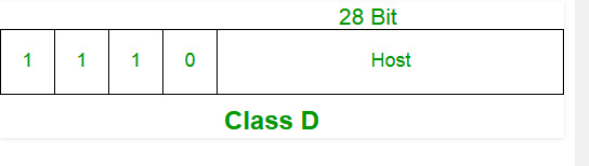
The higher order bits of the first octet of IP addresses of class C are always set to 110. The remaining 21 bits are used to determine network ID. The 8 bits of host ID is used to determine the host in any network. The default sub-net mask for class C is 255.255.255.x.



IP addresses belonging to class C ranges from 192.0.0.x – 223.255.255.x.

**Class D:**

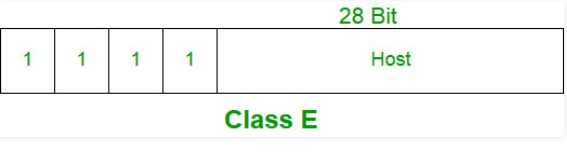
IP address belonging to class D are reserved for multi-casting. The higher order bits of the first octet of IP addresses belonging to class D are always set to 1110. The remaining bits are for the address that interested hosts recognize.



Class D does not posses any sub-net mask. IP addresses belonging to class D ranges from 224.0.0.0 – 239.255.255.255.

**Class E:**

IP addresses belonging to class E are reserved for experimental and research purposes. IP addresses of class E ranges from 240.0.0.0 – 255.255.255.254. This class doesn’t have any sub-net mask. The higher order bits of first octet of class E are always set to 1111.



**Problems with Classful Addressing:**

The problem with this classful addressing method is that millions of class A address are wasted, many of the class B address are wasted, whereas, number of addresses available in class C is so small that it cannot cater the needs of organizations. Class D addresses are used for multicast routing and are therefore available as a single block only. Class E addresses are reserved.

Since there are these problems, Classful networking was replaced by Classless Inter-Domain Routing (CIDR) in 1993.

## B1.3 A Validated C Program

# **Question No. 3**

**Solution to Question No. 3:**

## B2.1 Plagiarism rules and threshold

## B2.2 Pseudocode for checking plagiarized content

## B2.3 Validated C Program