classmate

Date ______
Page ______

Assignment - 18

Q.1 What is difference between static memory allocation and dynamic memory allocations?

→ • Static memory means the amount of memory to be given to the program is done at compile time.

• Dynamic memory means the amount of memory to be given to program is done at run time.

• Static memory allocation is considered as early binding.

• Dynamic memory allocation is considered as late binding.

• Static : Problem of memory wastage may ~~ou~~ occure.

• Dynamic : There no problem of wastage of of memory as memory is allocated at run time as per requirement.

• Static memory is allocated inside Heap.
• Dynamic memory is allocated inside stack.

Q.2 what are the advantages and disadvantages of Dynamic memory allocation over static memory allocation?

--> • Advantage : Dynamic memory allocation resolve problem of memory wastage as memory allocation is done at Run Time.

- No memory shortage.

- In static memory allocation the calcutions of memory done at compile time & memory allocated at Run time therefore memory wastage & shortage problem occures.

• when we allocate memory using dynamic approach same time memory is not given to program if there is no space on RAM.

- But in case of static the compilation error will occure in such case.

Q.3 which functions are used in c for dynamic memory allocation & deallocation.

→ ○ To allocate the memory there are three functions in c : (Dynamic)

1. malloc ()
2. calloc ()
3. realloc ()

○ To deallocate the dynamic memory :

1. free ()

Q.4 what functions / operators are used in c++ programming for dynamic memory allocation & deallocation.

○ To allocate memory in c++ programming (Dynamic) new () function is used.

○ To deallocate : delete () function is used.

for e.g → classname [*] pointer = new classname

delete (); // Free memory

Q.5 what is difference between malloc & calloc function.

→ ◦ malloc function is accepting only one parameter i.e no. of bytes that we want to allocate.

```
void * malloc (int size)   // prototype of
                              malloc
```

◦ The return value of malloc is void * [pointer] which indicates address of allocated memory.

```
for eg. →   isize = 0 ;   // user input
            int * ptr = NULL ; // NULL pointer

        ptr = (int*) malloc ( isize * sizeof (int) ;
```

- Typecasting required as we get address as return ∴ (int*).

◦ calloc function accepts two parameters.
- The first parameter is No. of element & second parameter is size of each element.

```
void * calloc (int no.of_elements,
               int size_of_each_element );
    // prototype of calloc
```

- Returns address

- Generally malloc used for Array.

for eg. ->

```
int isize = 0;   // input from user is
int * ptr = NULL // pointer       stored.

ptr = (int *) calloc (isize, sizeof
                       (int));
```

- Typecast (int *)

Q.6 Explain prototype of malloc Function

-> prototype : void * malloc (size);

- The return type of malloc () is a address. (void *)

- malloc () accepts only one parameter as (size) no. of bytes to be allocated.

for eg. ->

```
int isize = 0; //
int * ptr = NULL; //
                                   size of
ptr = (int *) malloc (isize * (int));
```

Q.7 why return value of malloc, calloc, realloc function is void.

→ ○ The malloc (), calloc () ,realloc () return the memory address to store that we uses pointer (*) .

- And we need different type of of pointers to store different types of datatype addresses which is dynamically assign , so for that we need need generic pointer void * therefore return value of all above functions is void.

Q.8 what are different uses of realloc function

→ ○ Realloc function is used to resize the allocated memory size.

○ Realloc function used to increase or decrease the size of already allocated memory.

○ realloc used for memory allocation as well as deallocation.

○ realloc :

//protope ty void * realloc (void *ptr, int new size) for eg. →

```
memory allocation using malloc
calloc : int * ptr = (int *) malloc (6* sizeof(int))
// Memory allocated 24 bytes
```

Increasing using realloc

```
ptr = (int*) realloc (ptr, 32)
// 8 bytes increase.
```

Decreasing using realloc

```
ptr = (int*) realloc (ptr, 16)
// 8 bytes decreas.
```

Q.9 what happens if first parameter of realloc function is NULL?

→ ◦ If first parameter of realloc () is NULL, Then we can used it is as malloc function

→ `void * reallo ( NULL * ptr , int new-size)`

becomes malloc function

`// void * reallo ( int new-size);`

Q10 what happens if second parameter of realloc function is 0?

→ ◦ If we give second parameter of realloc function 0 it working as free function & deallocate the memory

→ void * realloc (int * ptr. 0)