ASSIGNMENT 18

Q1] Explain difference between static memory allocation & dynamic memory allocation?

Points	Static	Dynamic
Def^{n}	Memory is allocated at compile time	Memory is allocated at runtime
Allocation method	Done by compiler automatically.	Done manually using ne or malloc()
Control	programmer has no control over memory size after allocation	programmer has full contr to allocate & deallocate memory.
Scope & Lifetime	Memory is created at compile time & destroyed automatically when the variable goes out of scope	Memory persists until explicitly deallocated using delete or free().
Efficiency	Faster as memory is allocated at compile time	Slower as memory is allocated at runtime.
Flexibility	Less flexible since the size of memory is fixed & cannot change	Highly flexible since memory size can be decided during execution.
Deallocation	Automatically handled by the compiler	Must be explicitly deallocated using delete or free()

Q2] What are the advantages & disadvantages of dynamic memory allocation over static memory allocation?

Advantages	Disadvantages
memory is allocated at runtime, offering flexibility	Requires manual deallocation, risking memory leaks.
Efficient use of memory by allocating only what's needed	Slower due to runtime allocation.
supports resizing & creation of complex data structures.	Increases program complexity. & risk of errors.
Lifetime of memory can be controlled explicitly.	Can lead to memory fragmentation

Q3] List down the functions & its syntaxes. Which are used in C Programming for dynamic memory allocation

1) malloc() {memory Allocation}

Allocates a block of memory of specified size in bytes

The memory is not initialised so it contains garbage values.

Syntax: void* malloc(size-t size);

size: no. of bytes to allocate

returns a pointer to allocated memory or NULL if the allocation fails.

Ex: int* ptr = (int*) malloc(5 * sizeof(int));

2) {Contiguous Allocation} calloc()

Allocates memory for an array of elements & initializes all bytes to 0.

Syntax: void * calloc(size-t num, size-t size);

num: no. of elements

size: size of each element in bytes

Returns a pointer to the allocated memory or NULL if the allocation fails

Ex: int* ptr = (int*)calloc(5,sizeof(int));

3) realloc() {Reallocation}

Resizes a previously allocated memory block to a new size
Can expand or shrink the memory block.

Syntax: void * realloc(void *ptr, size-t new-size);

ptr: pointer to previously allocated memory block

new-size:- new size in bytes

Returns a pointer to the newly allocated memory or NULL if the reallocation feels.

Ex ptr = (int*) realloc(ptr, 10 * sizeof(int));

4) free() {Free Allocated Memory}

Deallocates a previously allocated memory block & makes it available for reuse

Syntax: void free(void *ptr);

ptr: pointer to the memory block to be deallocated

Ex: free(ptr);

Q4] Which functions /operators are used in C++ for dynamic memory allocation.

1) new operator

Allocates memory dynamically from the heap for a single variable or an array

The memory is initialised to the default value for objects & unitialized for basic data types

Syntax: datatype* ptr = new datatype;

: datatype* arr = new datatype [size];

Ex: int * num = new int;

int * arr = new int [5];

2) Delete operator

Deallocates memory allocated by new & frees up the memory for reuse.

for arrays, the delete() operator is used.

Syntax: delete pointer;

delete[] arraypointer;

Ex:- delete num;

delete[] arr;

Q5] Write difference b/w malloc() & calloc().

Refer Q3

Q6] Explain the prototype of malloc function with example

Refer Q3

Q7] Why return value of malloc(), calloc() & realloc() is void* ?

A void* is a genric pointer type, meaning it can represent a pointer to any data type.

This allows malloc(), calloc(), realloc() to be flexible & allocate memory for any type of data without being restricted to a specific type.

Ex int* int_ptr= (int*) malloc (sizeof(int));
float* float-ptr= (float*) malloc (sizeof(float));

In both cases, malloc() returns a void*, which is then typecast to the appropriate type (int* or float*).

Q8] What are the different uses of realloc().

1) Used to Resize the allocated memory size

2) used to increase or decrease the size of already used

3) others: 1> Allocate fresh memory

2> Deallocate allocated memory

Ex int *ptr = (int*) malloc (5* size of(int));

1> Increase memory allocated:

int *ptr = (int*) realloc (ptr, 32);

2> Decrease memory allocated:

int *ptr= (int*) realloc (ptr, 16);

Q9] What will happen if we use 1st parameter of realloc(), as NULL

If 1st parameter of realloc() is NULL, it will work like malloc().

Ex: int *ptr = (int*) realloc (NULL,(30));

Q10] What will happen if 2nd parameter of realloc() is 0?

If 2nd parameter of realloc() is 0, it will work like free()

Ex int *ptr = (int*)realloc (ptr, (0));