Q1] Difference between malloc() & new, free & delete

Points	malloc()	new
Memory Allocation	Allocates raw memory	Allocates & calls constructor
Type	Function (Stdlib.h)	operator
Returns	Void * (needs typecasting)	properly typed pointer
Initialization	No Initialization	Calls constructor (for objects)
Failure Handling	Returns NULL	Throws std::bad_alloc &

Points	free()	delete
Type	Function (Stdlib.h)	Operator
Usage	used with malloc()	Used with new
Effect	Deallocates memory only	Calls destructor & deallocates memory
Pointer Type	Works on Void*	Works on object pointers

Q2] Program to allocate memory for 10 integers using malloc()

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int * arr = (int *)malloc(10 * sizeof(int));
if (arr == NULL){
printf("Memory Allocation failed\n");
return 1;
}
```

```
for(int i=0; i<10; i++) {
    arr[i] = i+1;
    printf("%d", arr[i]);
}
free(arr);
return 0;
}
```

Q8] Program to allocate memory for 10 integers using realloc().

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  int *arr = (int*)malloc(5 * sizeof(int));
  if(arr == NULL) {
    printf("Memory Allocation failed \n");
    return 1;
  }
  for(int i = 0; i<5; i++){
    arr[i] = i+1;
  }
arr = (int*)realloc(arr, 10 * sizeof(int));
for(int i=5; i<10; i++){
    arr[i] = i+1;
}
for(int i=0; i<10; i++){
printf("%d", arr[i]);
}
free(arr);
return 0;
}
```

Q4] What is a dangling pointer?

A dangling pointer is a pointer that continues to reference a memory location after the object it points to has been deallocated or goes out of scope.

This can occur, when an object is deleted but the pointer still holds the address of the non-invalid memory

To avoid dangling pointers, its a good practice to set the pointer to nullptr after deleting the memory

Q5] What is return value of malloc if memory manager is unable to allocate the memory?

The return value of malloc when it fails to allocate memory is NULL

if malloc successfully allocates the requested memory, it returns a pointer to the first byte of the allocated memory block

If it fails to allocate memory it returns NULL

Q6] Write a syntax which is used to allocate memory for N floats dynamically using malloc(), Accept the value of N from user at runtime.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  int N;
printf("Enter no. of floats: ");
scanf("%d", &N);
float *arr = (float*) malloc (N * sizeof(float));
if (arr == NULL) {
printf("Memory Allocation Failed\n");
} return 1;
```

```
for( int i=0; i<N; i++) {
  arr[i] = i * 1.1;
  printf("%0.2f", arr[i]);
}
free(arr);
return 0;
}
```

Q7] Allocate dynamic memory for array of 5 elements where each element is of below structure type.

```
struct hello {
    float f;
    int i;
};
```

```
#include<stdio.h>
#include<stdlib.h>
struct hello {
    float f;
    int i;
};
int main() {
  struct hello *arr = (struct hello*) malloc(5 * sizeof(struct hello));
  if (arr == NULL) {
  printf("Memory Allocation failed\n");
  return 1;
}
for (int i=0; i<5; i++) {
   arr[i].f = i * 1.1;
   arr[i].i = i+1;
```

```
printf("Element %d : f = %f :: = %d \n", i, arr[i].f,
arr[i].i);
}
free(arr);
return 0;
}
```

88] Explain internal working of new operator in detail.

new operator internally calls malloc().
No need of typecasting incase of new operator.
No resize activity allowed in C++ ie no realloc()
Initialises memory by calling the constructor (if for a class object)
Returns a pointer to allocated memory.

89] What is need of typecasting for return value of malloc()?

malloc() returns a void* which is a generic pointer.
Typecasting is required to convert the void* into the appropriate data type

Q10] Explain working of below application, draw its diagramatic representation & predict its o/p.

size : stores the no. of elements

p : pointer to dynamically allocated memory

iCnt : Loop counter

The program prompts the user to enter the no. of elements & stores it in size.

Dynamic Memory Allocation

```
p = (int *)malloc(size * sizeof(int));
```

The program prompts the user to enter size no. of integers & stores them in allocated memory

The program prints the elements stored in the allocated memory

free(p); releases the allocated memory.

o/p :- Enter number of elements

9

please enter the elements

10

20

30

Address	Value
p[0]	10
p[1]	20
p[2]	30