PAGE No.
DATE 17/10/23

Assignment - 19

Q1] What is the difference bet. malloc and new, free & delete

Ans

1] There is just no such difference between malloc & new except difference of language of programing.

2] malloc is used for dynamical allocation of memory in C & new is also used for dynamic memory allocation in C++

3] The new operator internally calls the malloc function

4] Also there is no such difference between free & delete.

5] malloc can be used in C++ for dynamic memory allocation but malloc will not call the constructor internally

6] whereas in C++, due to object orientation, the new & delete will create the object & destroy the object with constructor & destructor

Q2] Write a syntax which is used to allocate memory for 10 integers dynamically using malloc.

Ans

```
int *ptr = (int*) malloc (10 * sizeof(int));
```

~~int *ptr = malloc (10 * sizeof(int~~

Q3] Write a syntax which is used to allocate memory for 10 integers dynamically using realloc.

Ans

```
int *ptr = (int*) realloc (NULL, 10 * sizeof(int));
```

PAGE No.
DATE / /

Q4] What is meant by dangling pointer?

Ans

When we dynamically allocate memory & even free it after use, the pointer is still pointing to the de-allocated memory. This pointer pointing to the de-allocated memory is called dangling pointer.

This dangling pointer could contain garbage value. This value can be address of some other process.

So when we dereference the dangling pointer the operating system will kill that process giving segmentation fault.

Segmentation fault is when a process tries to access contents outside of its address space allocated by the operating system.

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

Ans If malloc(), calloc() or realloc() fails to allocate memory then they return NULL.
This is considered as failure.

PAGE No. / DATE / /

Q6] Write a syntax which is used to allocate memory for N floats dynamically using malloc. Accept the value of N from user at run-time.

Ans

```
printf("Enter size");
scanf("%d", &N);

float *ptr = (float*) malloc(N * sizeof(float));
```

Q7] Allocate dynamic memory for array of 5 elements where each element in below structure

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

Ans `struct hello*ptr = (struct hello*) malloc(5 * sizeof(struct hello));`

~~for(iCnt=0; iCnt<5; iCnt++~~

Q8] Explain the internal working of new operator in detail.

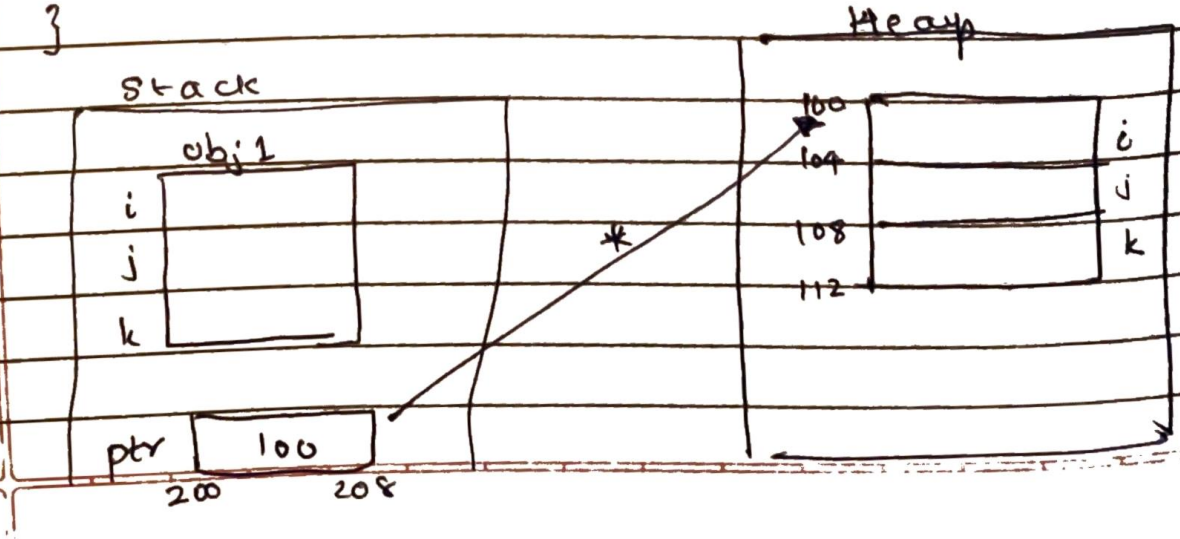
Ans.] new is an operator in C++ used to allocate memory dynamically on the heap

] The new operator internally calls the malloc function

] so there is no need of typecasting in case of new operator

Example

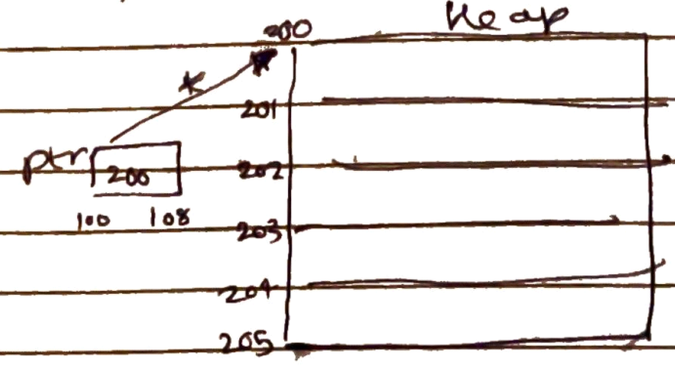
```
class Demo
{
  public:
   Demo()
   {
     cout<< "Inside constructor" <<endl;
   }
  ~Demo()
   {
    cout <<"Inside Destructor " <<endl;
   }
  void Display (){
      cout<<" Inside Display" <<endl;
};
int main ()
{
    Demo obj;
    obj1.display();

    Demo* ptr = new Demo;
     ptr -> display ()
     delete ptr ;
}
```



PAGE No.
DATE / /

1] we allocate dynamic memory for an object inside heap with the help of new
2] We can also allocate the dynamic memory by using malloc function, but the malloc function will not call the constructor internally.
3] Inshort new & delet will create & destroy the object along with the call of constructor & destructor

Q9] What is the need of typecasting for the return value of malloc function?
Ans The return value of malloc is void* which indicates the address of allocated memory After getting address we have to typecast depending on our requirement.
By Typecasting we divide the entirely allocated memory with return type void* into sections according to the casting type.
If we want to ~~divide~~ allocate memory for 5 Characters then we use
char *ptr = (char*) malloc (5 x sizeof(char));
this divides the allocated memory according to size of char i.e 1 bytes



PAGE No.
DATE / /

Q110] Explaining working of the below application, draw its diagrammatic representation & print its output.

Ans

```
#include <stdio.h>
#include<stdlib.h>

int main()
{
   int size = 0;
   int *p = NULL
   int iCnt = 0;

   printf(" Enter the number of elements \n");
   scanf(" %d", &size);

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

   printf(" Please enter elements");
   for (iCnt = 0; iCnt < size; iCnt++);
   {
      scanf(" %d", &p[iCnt]);
   }
   printf(" Entered elements are");

   for (iCnt=0; iCut < size; iCnt++)
   {
      printf(" %d", p[iCnt]);
   }
   free (p);
   return 0;
}
```

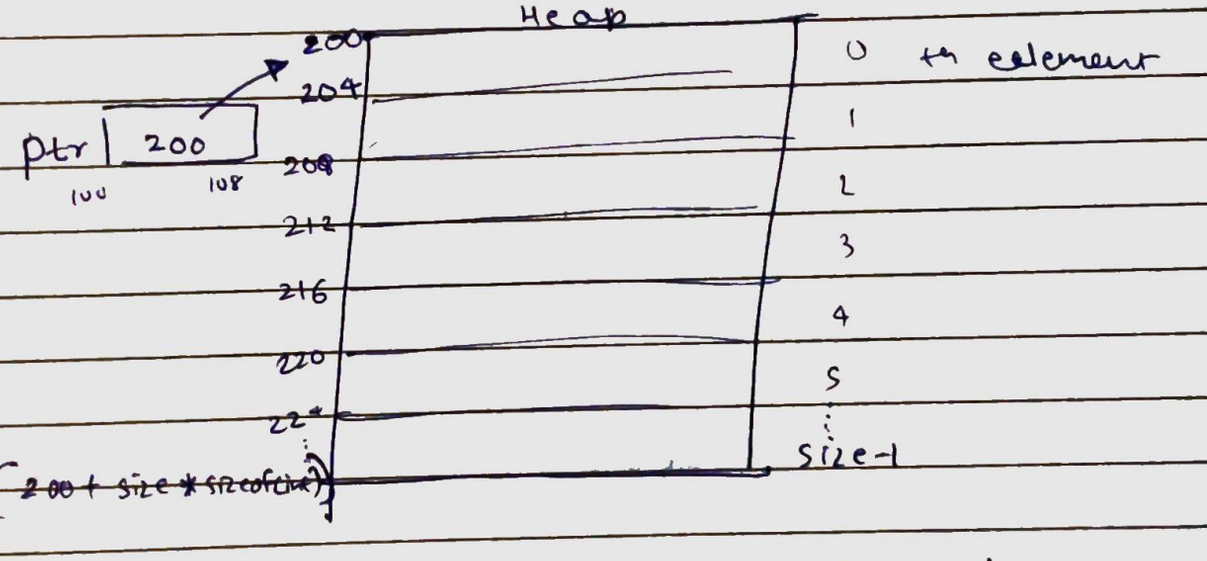
PAGE No.
DATE / /

1] In the application we include two header files <stdio.h> used for standard input & output functions, <stdlib.h> used for dynamic memory allocation functions (malloc) & (free)

2] We initialise ~~size~~ p (pointer) to NULL, to avoid segmentation faults

3] We initialise the loop counter iCnt = 0 & run the loop ~~till~~ incrementing iCnt ~~reaches~~ till it reaches the size (i.e no. of elements) given by user.

6] For each iteration in loop we store elements ~~ele~~ entered by user in the heap



7] The number enter by user gets stored inside heap, where iCnt = index of pointer. ~~array~~

8] This all is possible due to dynamically allocated memory

9] Similarly we print the elements stored insid heap with same number of iterations.