Wild Pointer , NULL Pointer , Generic Pointer and Dangling Pointer in C(same is applicable for C++)

Wild pointer:

A pointer in c which has not been initialized is known as wild pointer.

Example:

```
# include<stdio.h>
int main(){
int *ptr;
printf("%u\n",ptr);
printf("%d",*ptr);
return 0;
}
```

Output:
Any address
Garbage value

NULL pointer:

NULL pointer is a pointer which is pointing to nothing. NULL pointer points the base address of segment.

Examples of NULL pointer:
1. int *ptr=(char *)0;
2. float *ptr=(float *)0;
3. char *ptr=(char *)0;
4. double *ptr=(double *)0;
5. char *ptr='\0';
6. int *ptr=NULL;

We cannot copy any thing in the NULL pointer.

Example:

What will be output of following c program?

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

int main(){
char *str=NULL;
strcpy(str,"Everybrickmatters");
printf("%s",str);
return 0;
}
```

Output: (null)

Difference between NULL Pointer and Wild Pointer

There is difference between the NULL pointer and wild pointer. Null pointer points the base address of segment while wild pointer doesn’t point any specific memory location.

Generic Pointer

Void pointer in c is known as generic pointer. Literal meaning of generic pointer is a pointer which can point type of data.

Example:

```
void *ptr;
```
Here ptr is generic pointer.

Important points about generic pointer in c?

1. We cannot dereference generic pointer.

```
#include<stdio.h>
#include <malloc.h>
int main(){
void *ptr;
printf("%d",*ptr);
return 0;
}
```

Output: Compiler error

2. We can find the size of generic pointer using sizeof operator.

```
#include <string.h>
#include<stdio.h>
int main(){
void *ptr;
printf("%d",sizeof(ptr));
return 0;
}
```

Output: 2
Explanation: Size of any type of near pointer in c is two byte.

3. Generic pointer can hold any type of pointers like char pointer, struct pointer, array of pointer etc without any typecasting.

Example:

```
#include<stdio.h>
int main(){
char c='A';
int i=4;
void *p;
char *q=&c;
int *r=&i;
p=q;
printf("%c",*(char *)p);
p=r;
printf("%d",*(int *)p);
return 0;
}
```

Output: A4

4. Any type of pointer can hold generic pointer without any typecasting.

5. Generic pointers are used when we want to return such pointer which is applicable to all types of pointers. For example return type of malloc function is generic pointer because it can dynamically allocate the memory space to stores integer, float, structure etc. hence we type cast its return type to appropriate pointer type.

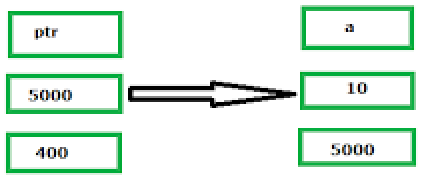
Examples:

```
1.char *c;
  c=(char *)malloc(sizeof(char));

2.double *d;
  d=(double *)malloc(sizeof(double));

3.Struct student{
  char *name;
  int roll;
  };
  Struct student *stu;
  Stu=(struct student *)malloc(sizeof(struct student));
```

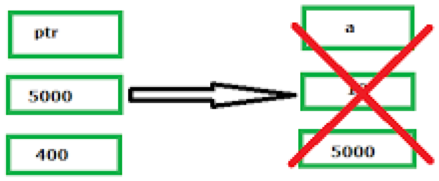
Dangling pointer:

If any pointer is pointing the memory address of any variable but after some variable has deleted from that memory location while pointer is still pointing such memory location. Such pointer is known as dangling pointer and this problem is known as dangling pointer problem.

Initially ptr points to a .



Now a has been deleted from the memory but ptr is still pointing to the same . This is dangling pointer .



See the following example of dangling pointer:

```
#include<stdio.h>

int *call();

void main(){
int *ptr;
ptr=call();
fflush(stdin);
printf("%d",*ptr);
}

int * call(){
int x=25;
++x;
return &x;
}
```

Output: Garbage value

How to solve the problem of Dangling Pointer ?
Solution of this problem: Make the variable x is as static variable.
In other word we can say a pointer whose pointing object has been deleted is called dangling pointer.

See the following example :

```
#include<stdio.h>

int *call();

void main(){
int *ptr;
ptr=call();
fflush(stdin);
printf("%d",*ptr);
}
```

```
int * call(){
static int x=25;
++x;
return &x;
}
```

```
Output: 26
```