1. Find which tool in compiler toolchain will give error in below code

```
int main()
{
	int a = 10;
	int b = 20;
	res = a + b;
	return 0;
}
int res;
```

2. Find which tool in compiler toolchain will give error in below code

```
extern int res;
int main()
{
	int a = 10;
	int b = 20;
	res = a + b;
	return 0;
}
```

3. Analyse the file generated by preprocessor tool

```
#define mask(x, bit)   x & ~(1<<bit)
int main()
{
	int a = 10;
	a = mask(a, 3);
	return 0;
}
```

4. Find which tool in compiler toolchain will give error in below code

```
#define 0max(x, y)     x>y?x:y
int main()
{
	int a = 10;
	int b = 20;
	res = 0max(a, b);
	return 0;
}
```

5. Analyse how static local variable information is stored in assemble file

```
int main() {
	static int a = 10;
	static double d;
	static char c = 'A';
	printf("a = %d, d = %.2lf, c = %c");
	return 0;
}
```

6. Analyse how global local variable information is stored in assemble file

```
int main() {
	static int a = 10;
	static double d;
	static char c = 'A';
	printf("a = %d, d = %.2lf, c = %c");
	return 0;
}
```

7. Write assembly equivalent for the following c code

```
int main()
{
	int a = 10, *p;
	p = &a;

	return 0;
}
```

8. Write assembly equivalent for the following c code

```
void add(int x, int y, int *ret)
{
	*ret = x+ y;
}

int main()
{
	int a = 10, b = 20, c;
	add(a, b, &c);
	printf("c = %d\n", c);
	return 0;
}
```

9. Create a static and dynamic libraries using the following source codes, and test the libraries

add.c

```
int add(int x, int y)
{
        return x+y;
}
```

sub.c

```
int sub(int x, int y)
{
        return x-y;
}
```

mul.c

```
int mul(int x, int y)
{
        return x*y;
}
```

div.c

```
int div(int x, int y)
{
        return x*y;
}
```

10. Write a test case to load the above dynamically created library at run time and invoke all library functions

11. Create the following source files

add.c

```
int add(int x, int y)
{
        return x+y;
}
```

sub.c

```
int sub(int x, int y)
{
    if(max(x, y))
        return x-y;
    else
        return y-x;
}
```

max.c

```
int max(int x, int y)
{
        return x>y;
}
```

min.c

```
int min(int x, int y)
{
        return x<y;
}
```

create **libari.so** using add.c, sub.c and **libm.so** using max.c, min.c

- **a)** Write a test cases to load **libari.so** library at run time using **RTLD_LAZY,** invoke ***sub*** function. if error comes resolve it.
- **b)** Write a test cases to load **libari.so** library at run time using **RTLD_NOW,** invoke ***sub*** function. if error comes resolve it.

12. Write a simple c program and analyse the address space of the program to check whether heap segment exist or not?

13. Write a simple c program which allocates 1024 bytes of dynamic memory using malloc and analyse the address space to find the total heap segment allocated.

14. Write a simple c program which allocates 1024 bytes of dynamic memory and populate the memory with 1, 2, 3, ……., 256.

15. Allocate 1024 bytes of dynamic memory using following calls
- a) brk
- b) sbrk

c) mmap
d) malloc
e) Deallocate the allocated dynamic memories

16) Allocate 8 bytes of dynamic memory using malloc and store two integers.
free the dynamic memory and access the freed memory, if no error explain why?

17. Allocate 8 bytes of dynamic memory using mmap and store two integers.
free the dynamic memory and access the freed memory, if error explain why?

18. Allocate 4096 bytes of memory using malloc, and find which system call is used by library to allocate memory.

19. Allocate 138k bytes of memory using malloc, and find which system call is used by library to allocate memory.

20. Which API should be used to allocate dynamic memory so that when allocate dynamic memory is released back to system.

21. Write a program which will print status of heap segment before dynamic memory allocation and after dynamic memory allocation.

22. write a program to find unused heap and release any memory above 4096 bytes.