# C Fundamentals Objective: Sheet 2

Q1.
```
int main()
  {
    short int i = 0;
    for(i<=5 && i>= 1;
    i++; i>0)
    printf("%u ",i);
    return 0;
  }
```
a) 1 ..... 65535 b) Expression syntax error c) No output d) 0 1 2 3 4 5

Q2.
```
#include<stdio.h>
  int main()
  {
    char arr[100];
    printf("%d",scanf("%s",arr)); //scanf gets input "GraphicEra"
    return 1;
  }
```
a) 10 b) 1 c) 1 d) 100

Q3.
```
int main()
  {
    static int i =5;
    if(i)
    {
      main();
      printf("%d", i);
    }
  }
```
a) 1 2 3 4 b) 4 3 2 1 c) 0 0 0 0 d) Compiler Error

Q4.
```
int main()
  {
    int i;
    i = 1,2,3,4;
    printf("%d", i);
    return 0;
  }
```
a) 1 b) 3 c) Garbage Value d) Compiler Error

Q5.
```
int main()
  {
    printf("%d", 2<<2+3<<3);
    return 0;
  }
```
a) 32 b) 88 c) 512 d) 0

Q6.
```
int main()
  {
    printf( 7 + "GraphicEraUniversity");
    return 0;
  }
```
a) 27 b) GraphicEraUniversity c) EraUniversity d) Compilation Error

Q7.
```
int main()
```

```
{
  int i = 12;
  int j = sizeof(i++);
  printf("%d, %d", i, j);
  return 0;
}
```

a) 12, 4 b) 0, 4 c) 13, 4 d) Compiler Error

Q8.
```
int main()
{
  int x = 0;
  int y = (~x == 1);
  printf("%d", y);
  return 0;
}
```

a) 0 b) 1 c) Compiler Error d) A bog of negative number

Q9.
```
int main()
{
  int a[5] = {1,2,3,4,5};
  int *ptr = (int*)(&a+1);
  printf("%d %d", *(a+1), *(ptr1));
  return 0;
}
```

a) 2 5 b) Grabage Value c) Compilation Error d) Segmentation Fault

Q10.
```
int f()
{
  static int n = 16;
  return n;
}
int main()
{
  for( f() ; f() ; f())
    printf("%d", f());
  return 0;
}
```

a) 15 12 8 5 2 b) 14 11 8 5 2 c) 13 10 7 4 1 d) Infinite Loop

Q11.
```
int main()
{
  enum status { pass, fail, atkt};
  enum status stud1, stud2, stud3;
  stud1 = pass;
  stud2 = atkt;
  stud3 = fail;
  printf("%d, %d, %d\n", stud1, stud2, stud3);
  return 0;
}
```

a) 0, 1, 2 b) 0, 2, 1 c) 1, 2, 3 d) 1, 3, 2

Q12.
```
void fun(char** str_ref)
{
  str_ref++;
}
int main()
```

```
{
  char *str = (void *)malloc(100*sizeof(char));
  strcpy(str, "GraphicEra");
  fun(&str);
  puts(str);
  free(str);
  return 0;
}
```

a) GraphicEra b) raphicEra c) Garbage Value d) Compiler Error

Q13.
```
int main()
{
  int x = 3;
  if(x == 2); x = 0;
  if(x == 3) x++;
  else x += 2;
  printf("%d", x);
  return 0;
}
```

a) Compiler Error b) 0 c) 2 d) 4

Q14.
```
int main()
{
  char *s[] = { "graphic", "era" "university"};
  char **p;
  p = s;
  printf("%s", ++*p);
  printf("%s", *p++);
  printf("%s", ++*p);
  return 0;
}
```

a) era university b) raphic raphic s c) era niversity d) raphic graphic era

Q15.
```
int main()
{
  int (*ptr) (int) = fun;
  (*ptr) (3);
  return 0;
}
int fun(int n)
{
  for( ; n>0 ; n)
  printf("Graphic");
  return 0;
}
```

a) Graphic Graphic b) Graphic Graphic Graphic c) Compiler Error d) Runtime Error

Q16. Which of the following is/are true?
a) calloc() allocates the memory and also initializes the allocated memory to zero, while memory allocated using malloc() has random data.
b) calloc() takes two arguments, but malloc takes only one argument.
c) both calloc and malloc return 'void' pointer.
d) all of the above.

Q17.
```
int main()
```

```
{
  char str1[] = "GraphicEra";
  char str2[] = {'G','r','a','p','h','i','c','E','r','a'};
  int n1 = sizeof(str1)/sizeof(str1[0]);
  int n2 = sizeof(str2)/sizeof(str2[0]);
  printf("n1 = %d, n2 = %d", n1, n2);
  return 0;
}
```

a) n1 = 9, n2 = 9 b) n1 = 9, n2 = 10 c) n1 = 10, n2 = 9 d) n1 = 10, n2 = 10

Q18.
```
int main()
{
  char str[] = "GraphicEra";
  printf("%s %s %s\n", &str[5], &5[str], str+5);
  printf("%c %c %c\n", *(str+7), str[7], 7[str]);
  return 0;
}
```

a) Compiler Error b) Runtime Error c) Era Era Era E E E d) icEra icEra icEra E E E

Q19.
```
int main()
{
int i = strlen(“BLUE”) + strlen(“PURPLE”) / strlen(red) – strlen(green);
printf(“%d”, i);
}
```

a) 1 b) -2 c) -1.666 d) -1

Q20. What is the purpose of fflush() function?

a) flushes all streams and specified streams b) flushes only specified stream
c) flushes input/output buffer d) flushes file buffer

Q21.
```
void fun(int **p);
int main()
{
  int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 8, 7, 8, 9, 0};
  int *ptr;
  ptr = &a[0][0];
  fun(&ptr);
  return 0;
}
void fun(int **p)
{
  printf("%d\n", **p);
}
```

a) 4 b) 3 c) 2 d) 1

Q22.
```
int main()
{
  int a[10];
  printf(“%d”, ((a+9) + (a+1)));
  return 0;
}
```

a) Error:invalid pointer arithmetic b) 19 11 c) 10 10 d) None of the above

Q23. Which of the following is the correct way to access the last element of the array arr, if arr is declared as int arr[3][3][4]?

a) *(*( (arr + 2 ) + 2) + 4)
b) *(*( *(arr + 3 ) + 3) + 4)
c) *(*( *(arr + 2 ) + 3) + 4)
d) *(*( *(arr + 2 ) + 2) + 3)

Q24. How to declare an array of N pointers to functions returning pointers to functions returning pointers to characters?
a) Char *(*(*a[N])))  b) Char (*(*(*a[N])))  c) Char (*(*a[N]))  d) None of the above

Q25.
```
int fun(int a0
    {
      printf("%d", a);
      return 0;
    }
    int main()
    {
      fun;
      return 0;
    }
```
a) It'll result in compile error because foo is used without parentheses.
b) No compile error and some garbage value would be passed to foo function. This would make foo to be executed with output "garbage integer".
c) No compile error but foo function wouldn't be executed. The program wouldn't print anything.
d) No compile error and ZERO (i.e. 0) would be passed to foo function. This would make foo to be executed with output 0.

**Solution: 1)**a. **2)**b. **3)**c. **4)**b. **5)**c. **6)**c. **7)**a. **8)**a. **9)**a. **10)**b. **11)**b. **12)**a. **13)**c. **14)**b. **15)**c. **16)**d. **17)**c. **18)**d. **19)**a. **20)**a. **21)**d. **22)**a. **23)**d. **24)**a. **25)**c.