C 2018

a) C Language basicly used to do low level programing So it is a low level language. But C Language also used in high level programing in sometimes also, so C Language can be considered as a Mid-level Language.

b) * Register * Automatic
* external * static

(c) (i) Global variables

```
#include<stdio.h>
void AC();
int x;
void main()
{
}
void AC()
{
}
```

This variable x can be used in both main and A functions. If there are more function still variable x can be seen and used in any function.

(ii)

```
#include<stdio.h>
void B();
void main()
{
int y;
}
void B()
{
int z;
}
```

This variable y can used only inside the main function. That means It can be intilized and used (Print) only inside the main function

Variable z is also like variable y in the main function. It only can be seen in B function

(iii)

```
#include<stdio.h>
int x;
void main()
{ int y;}
int s;
void fun1(){
int l;}
int k;
```

	x	y	s	l	k
main	✓	✓	x	x	x
fun1	✓	✓	✓	✓	

visibility (scope)

(d) • Preprocessing

first stage of compiling ~~in the~~ when preprocessing preprocessor checks commands that have # These command form a simple macro language. In this stage is used to ~~inti~~ include files define macros and to conditionally omit code (expanded source code)

• Compilation

In this stage, the preprocessed code is translated to assembly instructions ~~sep~~ specific to the target processor architecture. (Assembly code)

• Assembly

During this stage an assembler is used to translate the assembly instruction to object code. The output ~~contains~~ consists of actual instructions to be run by the ~~target~~ processor (~~Assembly~~ object code)

• Linking

The object code generated in the assembly stage is composed of machine instructions that the processor understands but some pieces of the program are out of order or missing. To produce an excutable program, the existing pieces have to be rearranged and the missing ones filled in. This process called linking. (executable code)

Q 2

(a)

```
#include<stdio.h>
void main()
{
int y[11],i;
for(i=0 ; i< 11 ; i++)
    y[i] = i-5;

puts("Array content : ");
for(i=0 ; i< 11 ; i++)
   printf("%d ", y[i]);
puts(" ");
}
```

(b) i)

```
#include<stdio.h>
#define UPPER_VOWEL(x) (x-32)
void main()
{
char ch;
printf("Enter charo letter:");
scanf("%c",&ch);
printf("%c ", UPPER_VOWEL(ch));
}
```

(x) (x=='a' || x=='e' || x=='i' || x=='o' || x=='u'
? x-32 : x)

(ii)

```
#includ ....
#define UPPER....
void main(){
char chh[100];
for(
```

```
int i ;
for(i=0 ; i<100 ; i++)
    {
    if( chh[i] == NULL)
    break;
    printf("%c ", UPPER_VOWEL(chh[i]));
    }
}
```

Q3

(a)

```
3  ⟶ int
4  ⟶ void
5  ⟶ i = 0
12 ⟶ system(cls);
24 ⟶ 'c' , choise
31 ⟶ { delay();
32 ⟶   break;
37 ⟶ } ''
38 ⟶
43 ⟶ } ''
44 ⟶
55 ⟶ } ''
56 ⟶
61 ⟶ } ''
62 ⟶
65 ⟶ break stop = 1;
66 ⟶ break;
```

(b)

```
int getInput()
{
int x;
printf(" ... ");
scanf("%d", &x);
return x;
}

void delay() {{   int i,j;
  for(i=0;i<1000;i++)
  for(j=0;j<1000;j++)
  }
```

Q4

(a) Modular programming is the process of subdividing a computer program into separate sub-programs In the program there can be several functions that are doing diffrent kind of work. So the program code doesn't have to ~~use~~ use same code again & again. Only have to call the want function.

b)

```
#include<stido.h>
int A();
void main()
{
  int x=10;
  printf("%d", A());
}

int A()
{
  return 0;
}
```

→ prints 0...

In here function is called ~~by~~ but not by value

```
#include<stdio.h>
int B(int x);
void main()
{
  int y=10;
  printf("%d", B(y));
}

int B(int x)
{
  x++;
  return x;
}
```

→ prints 11

In this program B function is called by value (B(y)) y value y sent to the function and it returned a value.

(c)

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void getInput();
void countInterchange();
void dispNumList();
int arr[10];

void main()
{
getInput();
}

void getInput()
{
int i;
srand(time(NULL));
for(i=0;i<10;i++)
{
int value=rand%100;
if(value<10)
value*=10;
arr[i]=value;
}
dispNumList();
printf("Interchange---:%d",countInterchanges());
}
```

Q 4

(c): ~~Re~~ code ↓↓

```
~~Dis~~
void dispNumList()
{
    int i=0;
    for(i=0; i<10; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int countInterchanges()
{
    int i, j, count=0;
    for(i=0; i<~~=~~9; i++)
    {
        for(j=0; j<9-i; j++)
        {
            int temp;
            if(arr[j+1] > arr[j])
            {
                temp = arr[j+1];
                arr[j+1] = arr[j];
                arr[j] = temp;
                count++;
            }
        }
    }
    return count;
}
```

printf("Generated List...#:\n");