1	`void fun(const int* ptr, const int N){`
	`      for(int i=0; i<N; i++, ptr++) {`
	`            *ptr = 5;`
	`            cout << *ptr;`
	`      }`
	`}`
	
	`int main(){`
	`      int arr[4] = {1,2,3,4};`
	`      fun(arr, 4);`
	`      return 0;`
	`}`
		[2 marks]
		
		Error: const int *ptr is a read only pointer, cannot assign 5
		
		Output after removing const keyword
		5555
		
		We use the pointer to constant (const * datatype ptr) when we don't want the pointer to be able to change the value at the address it points

2	`char *findChar(char *str) {`
	`      char *ptr = str;`
	`      while (*ptr != 's')`
	`            ptr++;`
	`      return ptr;`
	`}`
	
	`int main(){`
	`      cout << findChar("mystring");`
	`      return 0;`
	`}`
		[2 marks]
		
		string
		
		strings are char arrays, the findChar function returns a pointer pointing at 's' in the string.
		Since this is a string, cout displays all the characters in the string from 's' till the null character

3	`char *findChar(char *str) {`
	`      char *ptr = str;`
	`      while (*ptr != 's')`
	`            ptr++;`
	`      return ptr;`
	`}`
	
	`int main(){`
	`      cout << *findChar("mystring");`
	`      return 0;`
	`}`
		[2 marks]
		
		s
		
		strings are char arrays, the findChar function returns a pointer pointing at 's' in the string.
		cout displays the *value* this pointer points at

4	`void print(const char* p){`
	`      for(int i = 0; i < strlen(p);){`
	`            cout<<p<<endl;`
	`            p++;`
	`      }`
	`}`
	`int main(){`
	`      char p[] ={'1','2','3','\0'};`
	`      print(p);`
	`      return 0;`
	`}`
		[3 marks]
		
		123
		23
		3
		
		Strings are char arrays will null pointer at the end.
		Pointer to constants can point at constants and non-constants. strlen(p) return 3 (null char is not counted).

5	`void fun3(int&a){`
	`      a++;`
	`    cout<<a;`
	`}`
	`void fun2(int &a){`
	`    fun3(++a);`
	`    cout<<a;`
		[4 marks]
		
		4444
		
		If there is a local and global variable with the same name, the local one is used by default.

	`}`
	`void fun1(int &a){`
	    `fun2(++a);`
	    `cout<<a;`
	`}`
	`int a=5;`
	`int main(){`
	    `int a = 1;`
	    `fun1(a);`
	    `cout<<a;`
	    `return 0;`
	`}`	

6	`int g_One=1;`
	`void func(int* pInt){`
	      `pInt=&g_One;`
	`}`
	`void func2(int*& rpInt){`
	      `rpInt=&g_One;`
	`}`
	`int main(){`
	      `int nvar=2;`
	      `int* pvar=&nvar;`
	      `func(pvar);`
	      `cout<<*pvar<<endl;`
	      `func2(pvar);`
	      `cout<<*pvar<<endl;`
	      `return 0;`
	`}`
		[2 marks]
		
		2
		
		1
		
		In func() pointer parameter is passed by value, change made inside function only remains till function scope
		
		In func2() pointer parameter is passed by reference, any change made inside the function is retained outside the function also.

7	`int main(){`
	    `char sstring[] = {'g', 'n', 'o', 'r',`
	`'w','\0'};`
	    `char* chp = sstring;`
	    `chp += 4;`
	    `for(int i=0;i<5;i++){`
	      `cout <<*(chp-i);`
	      `}`
	    `return 0;`
	`}`
		[3 marks]
		
		wrong
		
		cout statement prints the value at which the pointer (chp) points. This will only print one character at a time.

8	`int main(){`
	    `int data = 10;       //address 200`
	    `int * const what;    //address 300`
	      `cout<<what<<"\t"<<*what<<"\\"<<&what;`
	
	      `return 0;`
	`}`
		[2 marks]
		Error: a constant pointer, similar to constant variables, MUST be initialized when declared.
		
		int *const what = &data; //correction
		200 10\300
		A constant pointer points at the same address during the entire program execution.
		Not to be confused with “pointer to constant”

9		**[2 marks]** **0**

	`int main(){`
	`    int array[] = {1,2,3,4,5};`
	`    int *p = array;`
	`    cout<<(p++ == array+1);`
	`    return 0;`
	`}`
		Post-increment evaluated after ==
		
		Before increment p stores the address of the first element in the array
		
		array+1 is the address of the second element in the array.
		
		Address of element 1 is not equal to the address of element 2

10	`int main(){`
	`    const int x = 10;`
	`    int *q = &x;`
	`    int *const_ptr = q;`
	`    cout << *const_ptr << endl;`
	`      return 0;`
	`}`
		[2 marks]
		Error: A simple int * cannot point at a constant variable. Only a pointer to constant can.
		const int *q=&x; //correction
		const int * const_ptr =q; //correction
		10
		Both pointers q and const_ptr are pointing at the address of a constant variable, therefore both should be pointers to constant

11	`int main(){`
	`  int arr[5]={1,5,9,11,15,19};`
	`  int i;`
	`  for(i=0;i<5;i++)`
	`       cout<<arr[i]/4*arr[i]/2<<"\t";`
	`  return 0;`
	`}`
		[3 marks]
		Error: 6 elements are assigned to array of size 5
		int arr[6]={1,5,9,11,15,19}; //correction
		0 2 9 11 22

12	`int main(){`
	`   int list[10]={21,12,13,3,55,16};`
	`   int i;`
	`   for(i=0;i<5;i++)`
	`   {`
	`      int temp=list[i];`
	`      list[i]=list[9-i];`
	`      list[9-i]=temp;`
	`   }`
	`   for(i=0;i<10;i++)`
	`       cout<<list[i]<<"\t";`
	`       return 0;`
	
	`}`
		[3 marks]
		
		0 0 0 0 16 55 3 13 12 21
		
		When initializing an array using the initializer list, if the values in the initializer list are less than the array size, remaining elements are initialized as 0 for numeric arrays.

13	`int main()`
	`{`
	`    int i,j,Matrix[4][4]={1, 3, 6,2,5, 9,1,`
	`7,8 , 4, 5 ,3,4,5,6,9};`
	
	`    for(i=0,j=N-1 ; i<N ; i++,j--)`
	`    {`
	`       if (Matrix[i][j]%4==0)`
		[2 marks]
		Error: N is not declared in this scope
		int N = 4; //correction
		1 0 5 3 5 3

```
            cout<<Matrix[i][j]+1<<"  ";
         cout<<Matrix[i][j]-1<<"  ";
      }
      return 0;
  }
```

14

```
  int main()
  {
      int i,j,Matrix[3][3]={1,2,3,4,5,6,7,8,9};
      for(int i=0;i<3;i++)
      {
         for(int j=0;j<3;j++)
         {
             if(i==j)
                cout<<Matrix[i][j]<<” ”;
         }
      }
  }
```

[2 marks]

1 5 9

15

```
int main(){

 int i = 50,j = 1, x=0 ;

 do{
     i= ++j;
     x++;
 }while(x<5);

 cout<<i<<"  "<<j;

}
```

[2 marks]

6 6

Prefix increment evaluated before the assignment

16

```
int main(){

 for(int i=0;;){
     i++;
     cout<<i<<"  ";
     if(i==3)
        break;
 }

}
```

[2 marks]

1 2 3

Stopping condition and update can be skipped in the for loop header

17

```
int main(){

    int something = 1;
    for(int i = n ; i>=0; i--){
        something = something * i;

        if(i==2)
            continue;
        if(i<3)
            break;
```

[2 marks]

Error: n isn’t declared

int n = 2; //correction

2

	`    }`
	`    cout<<something;`
	`}`	

18	`int main()`
	`{`
	`    int i=0, j=1;`
	`    while(i<5)`
	`    {`
	`        while(j<5){`
	`            cout<<"* ";`
	`            j++;`
	`        }`
	`        cout<<endl;`
	`        i++; j=i;`
	`    }`
	`    return 0;`
	`}`
		[2 marks]
		
		`* * * *`
		`* * * *`
		`* * *`
		`* *`
		`*`

19	`int main()`
	`{`
	`    int i = 0, j=1, c=0;`
	`    while(j - ++i) {`
	`        c++;`
	`    }`
	`    cout<<"Executed "<<c<<" times\n";`
	`    return 0;`
	`}`
		[2 marks]
		
		`Executed 0 times`
		
		Prefix executed before anything else in the statement 1 – 1 = 0
		if the loop condition is 0 (false) the loop does not execute

20	`int main()`
	`{`
	`    switch(~(12|25))`
	`    {`
	`        case 0:`
	`            cout<<"Programing ";`
	`        case 1:`
	`            cout<<"Fundamentals!";`
	`            break;`
	`        case -12:`
	`        case 29:`
	`            cout<<"is";`
	`            break;`
	`        case -29:`
	`            cout<<"fun";`
	`            break;`
	`        default:`
	`            cout<<"None of the case is true";`
	`    }`
	`    return 0;`
	`}`
		None of the case is true
		
		Bitwise OR of 12 and 25 = 29
		
		Then bitwise NOT of 29 = -30

21	`int calculation(int n) {`
	`    if (n > 1) {`
	`        return n * (n - 1);`
	`    } else {`
		`result = 20`

	`        return 1;`
	`    }`
	`}`
	
	`int main() {`
	`    int n, result;`
	
	`    n=5;`
	
	`    result = calculation(n);`
	`    cout << "result = " << result;`
	`    return 0;`
	`}`	

22	`int main()`
	` {`
	`          const int UPPER = 7, LOWER = 6;`
	`          int num1, num2, num3 = 12, num4 = 3;`
	
	`          num1 = num3 < num4 ? LOWER: UPPER;`
	`          num2 = num4 > UPPER ? num3 : LOWER;`
	
	`          cout << num1 << " " << num2 << endl;`
	`          return 0;`
	` }`
		`7 6`

23	`int main()`
	`{`
	`    int limit = 10;`
	`    cout<<((limit++) && (++limit - 12)) ;`
	`}`
		`0`
		
		The logical && has two expressions, one on each side. First left one is evaluated and then right one.
		
		The left expression only has one increment. Limit becomes 11.
		Right expression has prefix so limit becomes 12 and then 12 is subtracted from it.
		12-12 = 0
		
		Even if one expression is false the whole AND condition is false.
		
		Note: adding brackets does not change the order in which the postfix or prefix are evaluated. E.g. a++ + b; and (a++) + b; work the **SAME** way.

24	`#include <iostream>`
	`using namespace std;`
	
	`int main()`
	`{`
		Error: break statement can only be placed in a loop or switch
		
		Remove break statement

```
    int n=10;
    {
        n=20;
        break;
        n=30;
    }
    cout<<n;
    return 0;
}
```

30

25

```
#include <iostream>
using namespace std;

void test(int a);
int main(){

    test(10);
}

void test(int b){

    a = 20;
    b = 30;
    cout<<"a + b = "<< a * b;

}
```

Error: Variable a is not defined in test function.

Not an error but prototype should not have variable name

int a=20; //correction

600

26

```
#include <iostream>
using namespace std;


int do_something(int);

int main(){

    cout<<do_something(5);

}

int do_something(int n){

    int something = 1;
    for(int i = n ; i>=0; i--){
        something = something * i;

        if(i==2)
            continue;
        if(i<3)
            break;
    }
    cout<< something;
    exit(0);
    return 1;

}
```

120

27	`#include <iostream>`
	`using namespace std;`
	
	`int main(){`
	
	`   int a, b = 0;`
	`   if(a=a+b)`
	`       a = 2 * ++b + a++;`
	`        switch(a){`
	`            case 2:`
	`               b = 2 * a;`
	`            default:`
	`               b = (true ? (a > 0 ? 10 : 20 ) :`
	`30);`
	`               break;`
	`            case 0:`
	`                b = (a > 0 ? 1 : 2 );`
	`            case 3:`
	`                b = a + 1;`
	`                break;`
	`        }`
	`    cout<<a<<"  "<<b;`
	`}`
		0 1
		
		Assignment statement in if assigns the value 0 to a. If statement does not execute if the condition is 0 (false).
		Value of a is 0, case 0 is executed, but since there is no break after it, case 3 is also executed.

28	`#include <iostream>`
	`using namespace std;`
	
	`int main(){`
	`    int a = 2, b = 2, c =3, d =4;`
	`    a = a > b ? b : c > d ? c : d;`
	`    cout << a << endl;`
	`}`
		4

29	`#include <iostream>`
	`using namespace std;`
	
	`int main()`
	`{`
	` int testVal = 0;`
	`while (testVal++ < 10)`
	` {`
	` if (testVal == 4)`
	` continue;`
	` testVal = testVal+1;`
	` cout << testVal << " ";`
	` }`
	` return 0;`
	`}`
		[2 marks]
		
		2 4 6 8 10
		
		Post-increment, happens after comparison

30	`#include <iostream>`
	`using namespace std;`
	`int func (int);`
	`int main()`
	`{`
	` int x = 2, y = 3, z = 4;`
	` cout << “values of x, y, z before function`
	`calls`
	` are : “ << x << “ , “ << y << “ , “ << z<<endl;`
		[3 marks]
		
		Error: ‘i’ was not declared in this scope.
		
		for(int I=1, I<=x; I++) //correction
		
		values of x, y, z before function calls are : 2 , 3 , 4

	int x_value = func (x);
	int y_value = func (y);
	int z_value = func (z);
	cout << “values of x, y, z after function calls are : “ << x_value << “ , “ << y_value << “ , “ << z_value<<endl;
	return 0;
	}
	int func (int x = 6)
	{
	int temp=1;
	for(int I = 1; I <= x; i++)
	temp *= I;
	x = temp;
	return x;
	}
		values of x, y, z after function calls are : 2 , 6 , 24

31	#include <iostream>
	using namespace std;
	
	int main()
	{ int x = 9, y = 11;
	if (x < 10);
	cout << "@@@" << endl;
	if (y > 10)
	if (y> x);
	cout << "!!!" << endl;
	if (x==y)
	cout << "***" << endl;
	else
	cout << "###" << endl;
	cout << "$$$" << endl;
	return 0;
	}
		[2 marks]
		
		@@@
		!!!
		###
		$$$

32	int main(){
	
	double value = 92.8762;
	
	double *a, b;
	
	a = &value;
	b = a;
	cout<<*b;
	
	return 0;
	
	}
		Error: cannot save address in a double variable. Address can only be saved in a pointer.
		
		double *a,*b; //correction
		
		92.8762
		
		Pointer must be of the same type as the variable it points to