Section C (60 Points)

1a. Reference counter, and Mark & Sweep algorithms are used in the management of heap memory, explain them.

b. How does functional side effect influence referential transparency?

c. Explain the concept of operator overloading and show how * is overloaded in both C and C++.

d. What is the *protocol* of a subprogram?

1a. Reference counter maintains a counter in every cell that store the number of pointers currently pointing at the cell.

Mark & Sweep: Every heap cell has a status bit which is initially set as garbage. Subsequently, all pointers are traced into heap, and reachable cells are marked as **not garbage**. Finally, all garbage cells are returned to the list of available cells.

b. Non-functional languages that utilize variables are open to functional side-effect depending on implementation. When a function execution permits the alteration of variable not provided into it as a parameter, there is an accompanying chance of similar programs providing different outputs. This effect is the lack of referential transparency.

c. The use of an operator for more than one purpose is called operator overloading. * can be referred to as the multiplication operator, as well as the pointer operator in C and C++. Thus, providing possible compiler confusion for an instance when the left-hand side operand is missing in a multiplication expression.

d. The protocol is a subprogram's parameter profile (number, order, and types of its parameters) and, if it is a function, its return type.

2a. Provide the next 6 configurations of the LR parser over input string *id+id*id$*.

b. What happens if a parse falls on the circled cell (11, id)?

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5		S4				1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

1. E -> E + T
2. E -> T
3. T -> T * F
4. T -> F
5. F -> (E)
6. F -> id

a.

0	id+id*id$	Shift 5
0id5	+id*id$	Reduce 6 (Goto [0,F])
0F3	+id*id$	Reduce 4 (Goto[0,T])
0T2	+id*id$	Reduce 2 (Goto[0,E])
0E1	+id*id$	Shift 6
0E1+6	id*id$	Shift 5

b. A parse error.

3. Consider the following simple program.

```
void main()
{ int y=33
  a(6)
    void a(int y)
    { b();}
    void b()
    { c();}
    void c()
    { print y;}
}
```

a. In order to obtain the value of y in function c, how many dynamic links must be traversed, assuming the language supports shallow access? What is the output of c?

function a receives param 6 into var y memory location,

links= 2   y=6

b. In order to obtain the value of y in function c, how many static links must be traversed, assuming the language supports deep access? What is the output of c?

links: 1

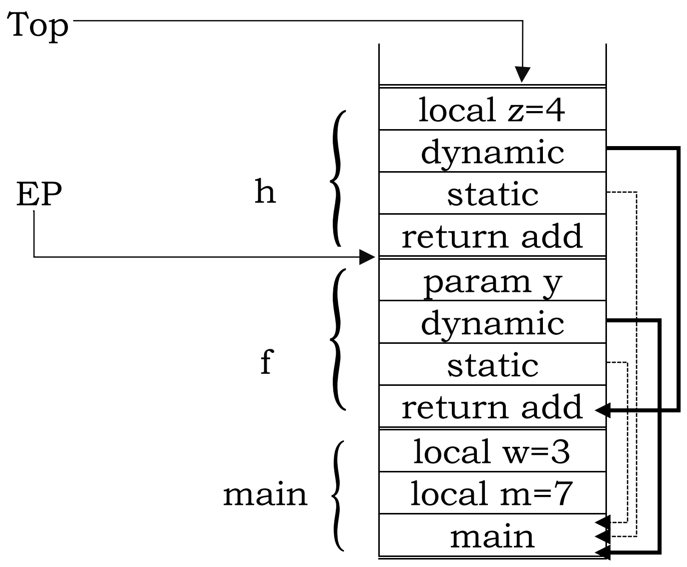
y=33

4. Consider the following program in a statically scoped language, where parameters are passed by value;

```
void main()
{ int w= 3, m=7;

    void g()
    {w+=2;
    print(w);}
    void h()
    {int z=4;
    print(z+w);}
    void f(int y)
    { if (y<5)
          h()
        else
        g();}
  f(w)}
```

Show the system stack content for the execution of the said program, including the activation record instance for subprogram calls. Do not forget to include all links, stack top marker, and an arrow to show the EP.



5. Consider the following function composition in Haskell.

```
member [] b = False

member(a:x) b = (a == b)|| member x b

squares = [n * n | n ← [0..10 `div` 2]]
```

After the execution of `member squares 9;`

a. What are the elements of squares

[0 1 4 9 16 25]

b. How many times does `member x b` evaluate?

[1 4 9 16 25] ---->first

[4 9 16 25] ---->second

[9 16 25] ---->third

It evaluates thrice!

c. What is the final value of `b`

True

6. Consider the following program in C syntax;

```
void fun (int first, int second)
  {first += first;
   second += second;}
void main()
  {int list[2] = {4, 3};
   fun(list[0], list[1]);}
```

For each of the following parameter-passing methods, what are the values

of the list array after execution?

a. Passed by value

{ 4,3}

b. Passed by reference

{ 8,6}

c. Passed by value-result

{ 8,6}