```
> #Q1
> proc1 := proc(L :: list)
  local L2, len, num;
  len := trunc( nops(L)/2 );
  L2 := [seq(0, i = 1 ..len)]; # initialize array with zeroes

  for num from 1 to len do
  L2[num] := L[num];
  end do;
  return L2;
  end proc;
```

```
proc1 := proc(L::list)                                                (1)
  local L2, len, num;
  len := trunc(1/2 * nops(L));
  L2 := [seq(0, i = 1 ..len)];
  for num to len do L2[num] := L[num] end do;
  return L2
end proc
```

```
> L := [1, 2, 3, 4, 5, 6, 7];
```

$$L := [1, 2, 3, 4, 5, 6, 7] \quad (2)$$

```
> proc1(L);
```

$$[1, 2, 3] \quad (3)$$

```
>
> #Q2
> #Done in class
>
> #Q3
> #similar example done in class
>
> #Q4
> #done in class
>
> #Q5
> partialSum := proc(L :: list, i :: integer)
  local num, sum;
  sum := 0;
  for num from 1 to i do
  sum := sum + L[num];
  end do;
  return sum;
  end proc;
```

```
partialSum := proc(L::list, i::integer)                               (4)
  local num, sum;
  sum := 0; for num to i do sum := sum + L[num] end do; return sum
end proc
```

```
> proc5 := proc(L :: list)
  local num, M;
  M := L;
  for num from 1 to nops(L) do
  M[num] := partialSum(L, num);
  end do;
  return M;
  end proc;
```

```
proc5 := proc(L::list)
    local num, M;
    M := L; for num to nops(L) do M[num] := partialSum(L, num) end do; return M
end proc
```

(5)

```
> L := [1, 2, 3, 4, 5];
```

$$L := [1, 2, 3, 4, 5]$$

(6)

```
> proc5(L);
```

$$[1, 3, 6, 10, 15]$$

(7)

```
> #Q6
> maxPartialSum := proc(L :: list)
  local num, currmax, currindex;
  #assume L is nonempty
  currmax := L[1];
  currindex := 1;
  for num from 2 to nops(L) do
  if partialSum(L, num) > currmax then
  currmax := partialSum(L, num);#update currmax to hold the current max value
  currindex := num; #update currindex to index of current max
  end if;
  end do;
  return currindex;
  end proc;
```

```
maxPartialSum := proc(L::list)
    local num, currmax, currindex;
    currmax := L[1];
    currindex := 1;
    for num to nops(L) do
        if currmax < partialSum(L, num) then
            currmax := partialSum(L, num); currindex := num
        end if
    end do;
    return currindex
end proc
```

(8)

```
> L := [1, 2, 3, 4]
```

$$L := [1, 2, 3, 4]$$

(9)

```
> maxPartialSum(L);
```

$$4$$

(10)

```
> L := [1, -1, 2, -1];
```

(11)

$$L := [1, -1, 2, -1] \quad (11)$$

```
> maxPartialSum(L);
```

$$3 \quad (12)$$

```
> #Q7
> f := (x, n) -> x^n/n!;
```

$$f := (x, n) \to \frac{x^n}{n!} \quad (13)$$

```
> myexp :=proc(x :: float, n :: posint)
  local num, sum, L;
  L := [seq(0, i=1..n)];#initialize list
  #first fill up the array
  for num from 1 to n do
  L[num] := f(x, num);
  end do;
  #compute sum of the array
  sum := 1; #the first term of the series is 1
  for num from 1 to n do
  sum := sum + L[num];
  end do;
  end proc;
```

```
myexp := proc(x::float, n::posint)
  local num, sum, L;
  L := [seq(0, i=1..n)];
  for num to n do L[num] := f(x, num) end do;
  sum := 1;
  for num to n do
      sum := sum + L[num]
  end do
end proc
```

(14)

```
> myexp(3.1,-2);
Error, invalid input: myexp expects its 2nd argument, n, to be
of type posint, but received -2
> #try it on a value
> exp(3.1)
```

$$22.19795128 \quad (15)$$

```
> myexp(3.1, 2);
```

$$8.905000000 \quad (16)$$

```
> myexp(3.1, 10);
```

$$22.18944041 \quad (17)$$

```
> myexp(3.1, 20);
```

$$22.19795128 \quad (18)$$

```
>
```