Answer all questions.

1. Tom develops a music streaming platform. Users listen to the songs on a playlist. Tom designs the playlist as a queue with the following global variables:

Variable	Description
`A`	An array for implementing the queue with indices from 0 to 6
`first`	An integer variable that `A[first]` stores the first song on the playlist
`size`	An integer variable for storing the number of songs on the playlist

Assume that the initial contents of `A`, `first` and `size` are

i	0	1	2	3	4	5	6
A[i]	Hat	Fly	Bee	Run	Kitty	ABC	Stars

`first = 4` `size = 3`

The songs on the playlist listed in order are `Kitty`, `ABC` and `Stars`.

Tom writes the subprogram `deq`. If the playlist is not empty, `deq` will remove a song from the playlist. The pseudocode for `deq` is

```
deq
    if size = 0 then
        output 'The playlist is empty!'
    else
        first ← remainder of ((first + 1) ÷ 7)
        size ← size - 1
```

(a) (i) Assume that the initial contents of `A`, `first` and `size` are

i	0	1	2	3	4	5	6
A[i]	Bus	Fly	Run	Hart	Stars	ABC	Kitty

`first = 5` `size = 4`

Write down the values of `first` and `size` after executing `deq` each time.

	first	size
After executing `deq` for the first time	0	5
After executing `deq` for the second time	1	6

(3 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Tom writes another subprogram `enq(K)` to add a new song `K` to `A`.

For example, the initial contents of `A`, `first` and `size` are

i	0	1	2	3	4	5	6
A[i]	Bus	Smile	Eyes	Car	Bee	ABC	Kitty

`first = 4` `size = 3`

After executing `enq('Pot')`, the contents of `A`, `first` and `size` are

i	0	1	2	3	4	5	6
A[i]	Pot	Smile	Eyes	Car	Bee	ABC	Kitty

`first = 4` `size = 4`

When the playlist already has 7 songs and `enq(K)` is executed, `K` can still be added to the end of the playlist. However, the first song of the playlist will be removed.

Assume that the initial contents of `A`, `first` and `size` are

i	0	1	2	3	4	5	6
A[i]	Bus	Smile	Eyes	Car	Bee	ABC	Kitty

`first = 4` `size = 7`

After executing `enq('Pot')`, the contents of `A`, `first` and `size` become

i	0	1	2	3	4	5	6
A[i]	Bus	Smile	Eyes	Car	Pot	ABC	Kitty

`first = 5` `size = 7`

(ii) Complete the pseudocode for `enq` below.

```
enq(K)
    A[ i to 7 ] ← K
    if size < 7 then
        size ← K
    else
        first ← i + 1
```

(3 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

(iii) Tom writes a subprogram `playPL` that plays all the songs on the playlist. `playSong(A[curr])` is another subprogram that plays the song stored in `A[curr]`. Complete the pseudocode for `playPL` below.

```
playPL
    for i from 1 to [ N ] do
        curr ← first + i - 1
        if curr [ > 1 ] then
            curr ← [ first + i + 1 ]
        playSong(A[curr])
```

(3 marks)

(b) Tom uses another implementation of the queue with the following global variables:

Variable	Description
`Song`	An array for implementing the queue with indices from 0 to 6
`Next`	An array for storing the index of the next song in `Song`; `Next[i]` stores -1 if `Song[i]` is the last song on the playlist
`ft`	An integer variable for storing the index of the first song

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

In the following example, the songs listed in order are `Run`, `Bus`, `Stars` and `ABC`.

i	0	1	2	3	4	5	6
Song[i]	ABC	Bus	Run	Stars			
Next[i]	-1	3	1	0			

```
ft = 2
```

(i) Tom rewrites the subprogram `playPL`. Complete the pseudocode for `playPL` below.

```
playPL
    curr ← [ Next[i] ]
    while [ Song[i] = -1 ] do
        playSong(Song[curr])
        curr ← [ Song[i] ]
```

(3 marks)

(ii) Referring to the example above, a song `Smile` is added to the end of the playlist. Complete `Next` below.

i	0	1	2	3	4	5	6
Song[i]	ABC	Bus	Run	Stars	Smile		
Next[i]	-1	3	1	0	1		

```
ft = 2
```

(1 mark)

(iii) Tom writes a subprogram `insert(sg)` that adds a new song `sg` to the beginning of the playlist. Complete the pseudocode for `insert(sg)` below.

```
insert(sg)
    Song[new_pos] ← sg
    Next[new_pos] ← [ Song[i+1] ]
    ft ← [ 2 ]
```

(2 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Go on to the next page

2. Peter records the daily usage of his mobile phone in minutes. He stores the daily usage of `N` days in an array `T`. The values in `T` are sorted such that `T[1] ≤ T[2] ≤ ... ≤ T[N-1] ≤ T[N]`.

(a) Peter designs a subprogram `bsearch(tar)` that returns `pos` such that `T[pos] = tar` by using binary search. The pseudocode for `bsearch(tar)` is

Line number	Content
1	`bsearch(tar)`
2	`  left ← 1`
3	`  right ← N`
4	`  pos ← -1`
5	`  while (pos = -1) AND (left <= right) do`
6	`  mid ← integral part of ((left + right) ÷ 2)`
7	`  if (T[mid] = tar) then`
8	`  pos ← mid`
9	`  if (T[mid] < tar) then`
10	`  left ← mid + 1`
11	`  if (T[mid] > tar) then`
12	`  right ← mid - 1`
13	`  return pos`

Assume that `N = 7` and the content of `T` is

i	1	2	3	4	5	6	7
T[i]	45	50	60	75	140	240	300

(i) `bsearch(140)` is executed. What are the values of `left` and `right` after the first three passes of Line 6? Complete the following table.

	left	right
First pass	1	7
Second pass	4	7
Third pass	4	5

(2 marks)

(ii) `bsearch(tar)` returns `-1` for some values of `tar`. Why?

Because ~~it is not~~ the values are not satisfied to ~~the~~ tar.

(1 mark)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

(b) Peter designs `newsearch(tar)` by modifying `bsearch(tar)`. He modifies Line 5 and inserts a new statement on Line 9. The pseudocode for `newsearch(tar)` is

Line number	Content
1	`newsearch(tar)`
2	`  left ← 1`
3	`  right ← N`
4	`  pos ← -1`
5	`  while (left <= right) do`
6	`    mid ← integral part of ((left + right) ÷ 2)`
7	`    if (T[mid] = tar) then`
8	`      pos ← mid`
9	`      right ← mid - 1`
10	`    if (T[mid] < tar) then`
11	`      left ← mid + 1`
12	`    if (T[mid] > tar) then`
13	`      right ← mid - 1`
14	`  return pos`

Assume that `N = 7` and the content of `T` is

i	1	2	3	4	5	6	7
T[i]	45	45	150	150	150	240	300

(i) `newsearch(150)` is executed. What are the values of `left`, `right` and `pos` after the first three passes of Line 6? Complete the following table.

	left	right	pos
First pass	1	7	-1
Second pass	2	6	-1
Third pass	3	5	-1

(3 marks)

Go on to the next page

(ii) `newsearch(149)` will return `-1` on Line 14. What are the values of `left` and `right` just before the return?

`left:` 2 `right:` 3

(2 marks)

(iii) Peter modifies Line 14 of `newsearch(tar)`. It will return the smallest `k` such that `T[k] >= tar`. Complete Line 14 below.

(1) Line 14: `return` tar

(1 mark)

(2) `count(p)` is a subprogram that returns the number of days on which the daily usage of the mobile phone is `p` minutes or above. Complete the pseudocode for `count(p)` by using `newsearch` below.

```
count(p)
    return  p. ← newsearch(p).
```

(2 marks)

(c) (i) Peter chooses a compiler instead of an interpreter to write programs. Give **two** reasons to support his choice.

- lower cost
- lower risk of the program.

(2 marks)

(ii) Linker and loader are used in the execution of his programs. Describe the functions of linker and loader.

Linker: The program can open in the link.

Loader: The program pre-loaded before open the program.

(2 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

3. Alice manages a project with several sorting subprograms.

(a) Alice writes a sorting subprogram `MySort` where `C` is an array with `n` elements. The pseudocode for `MySort` is

Line number	Content
1	`MySort`
2	`  for i from 1 to n do`
3	`    for j from 1 to n-1 do`
4	`      if C[j] > C[j+1] then`
5	`        temp ← C[j+1]`
6	`        C[j+1] ← C[j]`
7	`        C[j] ← temp`
8	`    Output 'Outer loop: ', i`

(i) `MySort` is executed with the initial content of `C` below:

k	1	2	3	4
C[k]	36	29	18	55

Fill in the contents of `C[1]` and `C[2]` after the first two passes of Line 8.

	i	C[1]	C[2]
First pass	1	1	2
Second pass	2	2	3

(3 marks)

(ii) What type of sorting algorithm is `MySort`? grene

(1 mark)

Answers written in the margins will not be marked.

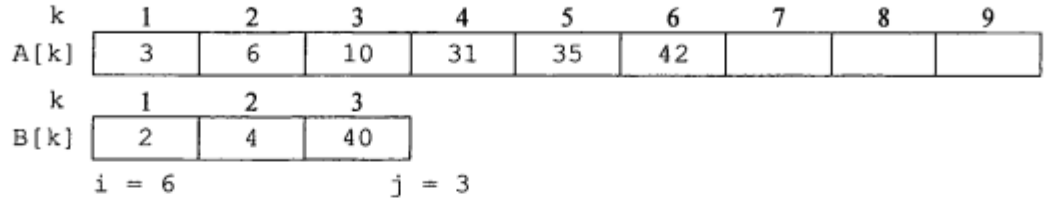
Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Go on to the next page

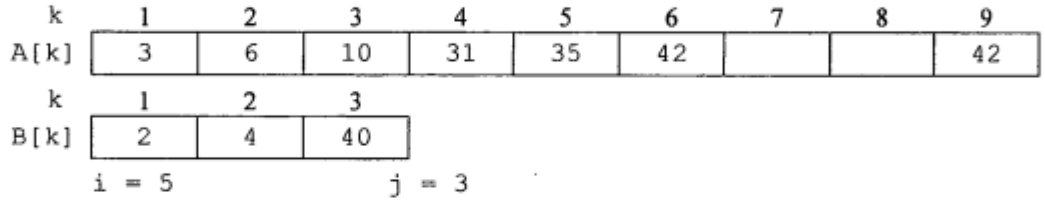
(b) Alice writes another sorting subprogram `MS` to merge two sorted arrays, `A` and `B`, and store it in `A`. `m` and `n` are the numbers of elements in `A` and `B` respectively. `i` and `j` are integer variables.

Below is an example with `m = 6` and `n = 3`. The contents of the variables before executing the loop are

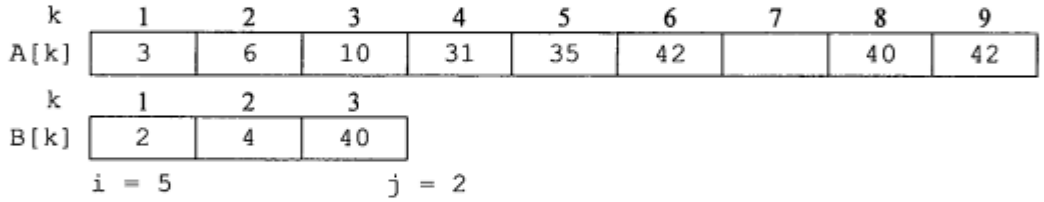


The contents of the variables after the first two passes of the loop are

The first pass:



The second pass:



Complete the pseudocode for `MS` below.

Line number	Content
1	`MS`
2	`    i ← m`
3	`    j ← n`
4	`    while` A > B `do`
5	`        if (i > 0) AND (A[i] > B[j]) then`
6	`            A[i+j] ←` A[k]
7	A[i] ← i - 1
8	`        else`
9	`            A[i+j] ←` A[j]
10	A[j] ← j - 1

(5 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

(c) There are 5 tasks below:

Task	Duration (day)	Depending on
A	3	B
B	3	-
C	4	B, D
D	2	-
E	1	C, D

(i) Complete the following Gantt Chart.

	Day									
	1	2	3	4	5	6	7	8	9	10
Task A										
Task B										
Task C										
Task D										
Task E										

(3 marks)

(ii) If Task C no longer depends on Task B, what is the minimum number of days needed to complete all tasks?

7 days.

(1 mark)

(iii) After Alice completes the programs in Task C, she will carry out a user acceptance test and a unit test. Which test should she carry out first? Justify your answer.

user acceptance test, because she should check the program is satisfied to the user needed before the unit text.

(2 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

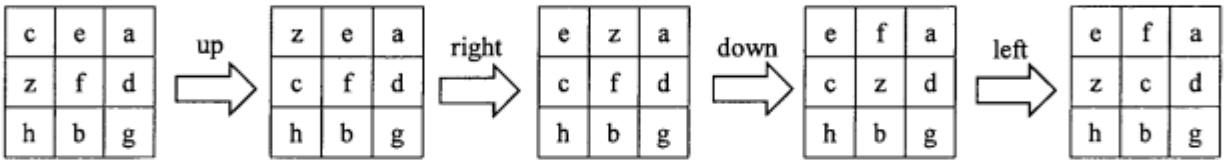
Go on to the next page

4. Eva writes a program to simulate a board game with 3×3 cells. Each cell contains a character a, b, c, d, e, f, g, h or z. She uses an array `BD` to represent the board. `BD[i,j]` stores the character in the corresponding cell on the board. `BD[1,1]` refers to the top left corner of the board. The character z is stored in `BD[x,y]` where `x` and `y` are variables. In the example below, `x = 2` and `y = 1`.

BD

i \ j	1	2	3
1	c	e	a
2	z	f	d
3	h	b	g

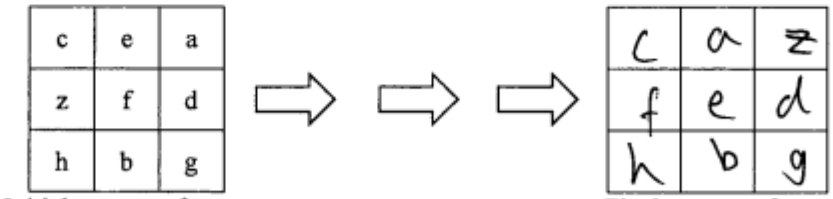
The player can swap the character z with one of its adjacent characters using the four swaps shown below: up, right, down and left.



Eva writes a subprogram `swapBD(dir)` to perform the swaps. The following table shows the corresponding value of `dir` for each swap.

swap	dir
up	0
right	1
down	2
left	3

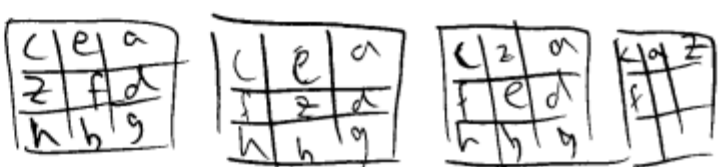
(a) Three swaps, `swapBD(1)`, `swapBD(0)` and `swapBD(1)`, are sequentially executed below. Fill in the final content of `BD`.



Initial content of `BD` Final content of `BD`

(2 marks)

Answers written in the margins will not be marked.

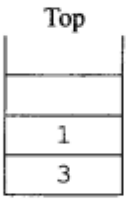


(b) Complete the pseudocode for part of `swapBD(dir)` below.

```
if dir = 0 then
   if x = 1 then output 'up: swap not successful'
   else
        BD[x,y] ← [swapBD(0)]
        [BD(x,y) ← ≠ ]
        x ← x - 1
if dir = 1 then
   if y = 3 then output 'right: swap not successful'
   else
        [B[x,y) ← swapBD(1).]
        BD[x,y+1] ← 'z'
        [y ← y+1]
```

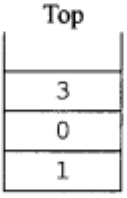
(4 marks)

After executing each swap, `dir` is pushed into a stack `History`. For example, after `swapBD(3)` and `swapBD(1)` are executed in sequence, the content of `History` is



(c) Eva writes a subprogram `resetBD` that resets `BD` to its initial content with the use of `History`.

(i) Assume that the content of `History` is



Write down the sequence of `swapBD` executed in `resetBD`.

`swapBD(` 1 `)` ⇨ `swapBD(` 0 `)` ⇨ `swapBD(` 3 `)`

(2 marks)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

To operate a stack `S`, Eva uses the following subprograms:

Subprogram	Description
`push(S,X)`	Adds an item `X` to `S` as its top element
`pop(S)`	Removes and returns the top element of `S`

(ii) Complete the pseudocode for `resetBD` below.

```
resetBD
  while History is not empty do
    topH ← pop(History)
    swapBD( push S , topH )
```

(2 marks)

(iii) The final contents of `BD` and `History` are

b	h	a
f	d	c
e	z	g

Final content of `BD`

Top

2
1
3
1
3
1
3

Final content of `History`

Eva finds that only one swap is needed to reset `BD` to its initial content. What is the swap?

`swapBD( 2 )`

(1 mark)

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.

(iv) Eva writes a subprogram `SimplifyH` to remove consecutive items in `History` that can be ignored for `resetBD`. `TmpStack` is a temporary stack.

Complete the pseudocode for `SimplifyH` below.

```
SimplifyH
  while History is not empty do
    if TmpStack is empty then
        push(TmpStack, [ pop(History) ])
    else
        topH ← pop(History)
        topT ← pop(TmpStack)
        if (difference between topH and topT) ≠ [ tar ] then
            push(TmpStack, [ topH ])
            push(TmpStack, [ topT ])
  while TmpStack is not empty do
    push(History, pop(TmpStack))
```

(4 marks)

END OF PAPER

Answers written in the margins will not be marked.

Answers written in the margins will not be marked.