



UNIVERSITI TUN HUSSEIN ONN MALAYSIA

**FINAL EXAMINATION
SEMESTER II
SESSION 2018/2019**

COURSE NAME : DATA STRUCTURE & ALGORITHM
COURSE CODE : BIT 10703
PROGRAMME CODE : BIT
EXAMINATION DATE : JUNE / JULY 2019
DURATION : 3 HOURS
INSTRUCTION : A) ANSWER **ALL** QUESTIONS
B) PLEASE WRITE YOUR ANSWERS
IN THIS QUESTION BOOKLET

THIS QUESTION PAPER CONSISTS OF **TEN (10)** PAGES

CONFIDENTIAL

TERBUKA

Q1 (a) Determine the output for the program in Figure Q1(a).

```
#include <stdio.h>

long testMe( long n );
int main()
{
    long result;

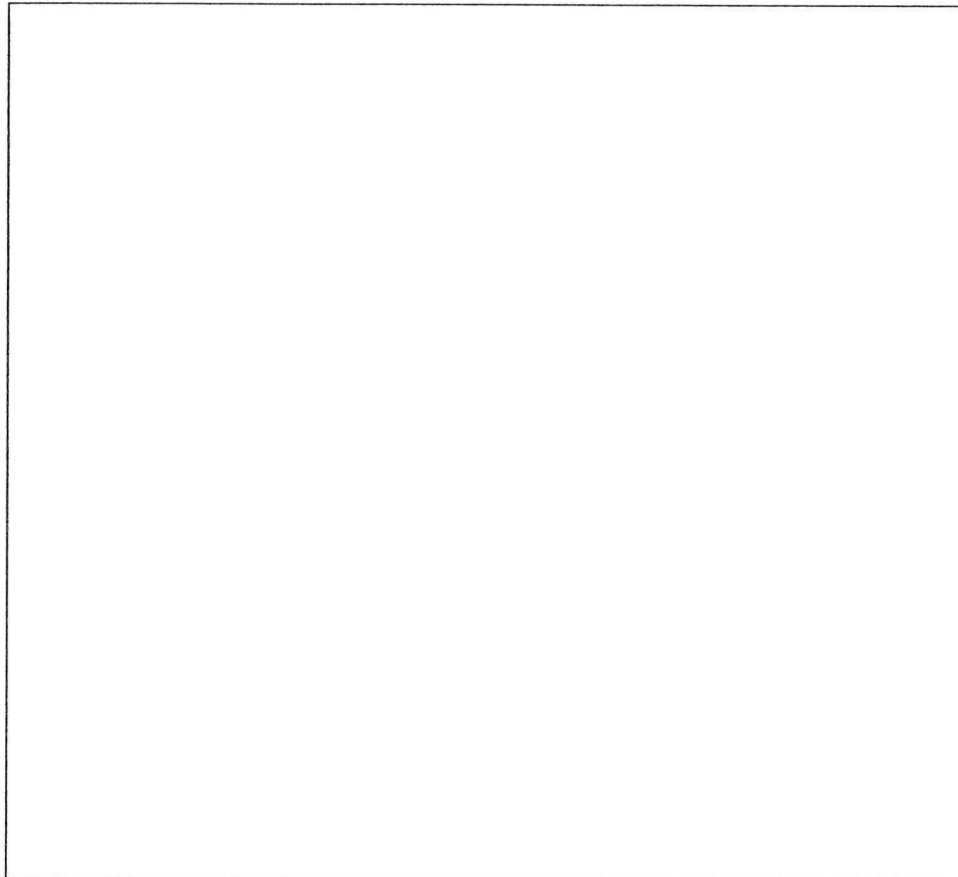
    result = testMe(5);
    printf( "\nResult = %ld\n", result );

    return 0;
}

long testMe( long n )
{
    if ( n == 0 || n == 1 ) {
        printf("  %ld", n);
        return n;
    }
    else {
        return testMe( n - 1 ) + testMe( n - 2 );
    }
}
```

Figure Q1(a)

Answer:



- (b) Write a code segment for Figure Q1(b) that will display even values only from the linked list.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int main(){

    struct NumNode{
        int value;
        struct NumNode *next;};

    struct NumNode *head = NULL, *pl;
    struct NumNode *curr;

    int i, num[SIZE] = {47, 34, 68, 13, 124};

    for(i=0; i<SIZE; i++)
    {
        pl = malloc(sizeof(struct NumNode));

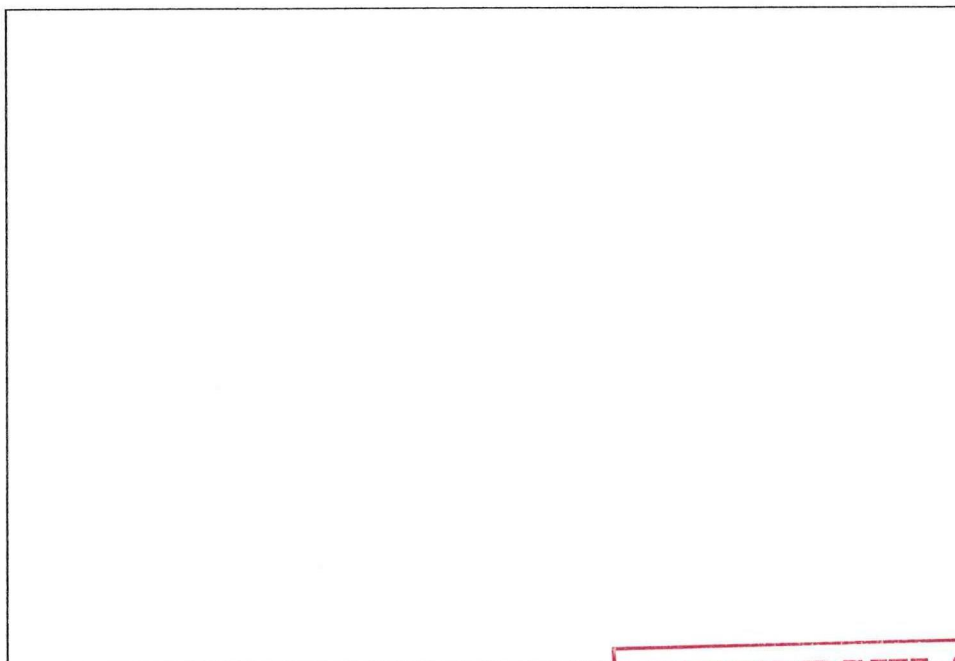
        pl->value = num[i];
        pl->next = NULL;

        if (head == NULL)
            head = pl;
        else
            { pl->next = head;
              head = pl;}
    }

    return 0;
}
```

Figure Q1(b)

Answer:



- (c) Write a function, named `findMax` for Figure Q1(c) that will return the maximum salary from the linked list.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 10

struct Staff{
    char name[20];
    double salary;};

typedef struct Staff Staff;

struct StaffNode{
    Staff staffData;
    struct StaffNode *next;};

int main(){
    int i;
    char staffName[20];
    double staffSalary;

    struct StaffNode *head = NULL, *p1;

    for(i=0; i<SIZE; i++)
    {
        printf("\nEnter name:");
        scanf("%s", staffName);
        printf("\nEnter salary:");
        scanf("%lf", &staffSalary);

        p1 = malloc(sizeof(struct StaffNode));

        strcpy(p1->staffData.name, staffName);
        p1->staffData.salary = staffSalary;
        p1->next = NULL;

        if (head == NULL)
            head = p1;
        else
            { p1->next = head;
              head = p1;}
    }

    return 0;
}
```

Figure Q1(c)**TERBUKA**

Answer:

(10 marks)

Q2 Answer **Q2(a)** based on the program given in **Figure Q1(c)**.

- (a) Write a code segment that will remove a node from the linked list according to stack operation.

Answer:

TERBUKA (10 marks)

- (b) Complete Figure Q2(b) with a code segment that will insert a new node to the linked list according to queue operation.

Answer:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 10

struct Staff{
    char name[20];
    double salary;};

typedef struct Staff Staff;

struct StaffNode{
    Staff staffData;
    struct StaffNode *next;};

int main(){
    int i;
    char staffName[20];
    double staffSalary;

    struct StaffNode *head = NULL, *p1;

    for(i=0; i<SIZE; i++)
    {
        printf("\nEnter name:");
        scanf("%s", staffName);
        printf("\nEnter salary:");
        scanf("%lf", &staffSalary);

        p1 = malloc(sizeof(struct StaffNode));

        strcpy(p1->staffData.name, staffName);
        p1->staffData.salary = staffSalary;
        p1->next = NULL;
    }
}
```

Answer:

```
    }//for
    return 0;
}
```

- Q3 (a)** Figure **Q3(a)** presents a program with a sorting function, called `SortData`. Determine the output for Figure **Q3(a)**.

```
#include <stdio.h>
#define SIZE 9

int main()
{
    int value[SIZE] =
{122,105,167,198,145,132,201,199,173};
    void function1(int[],int);

    SortData(value,SIZE);

    return 0;
}

void SortData(int list[],int size)
{
    int i,j,k,temp;

    for(i=0; i<(size-1); i++)
    {
        for(j=1; j<size; j++)
        {
            if (list[j]<list[j-1])
            {
                temp = list[j];
                list[j] = list[j-1];
                list[j-1] = temp;
            }
        }

        printf("\n value: ");
        for(k=0; k<size; k++)
            printf("  %d", list[k]);
    }
}
```

Figure Q3(a)**Answer:**

(8 marks)

TERBUKA

- (b) Name the sorting algorithm perform in Q3(a).

Answer:

(2 marks)

- a. Determine the output for Figure Q3(b).

```
int testQ3( int list[], int no, int x)
{
    int idx, test, left, right, mid;

    idx = -1;
    test = 0;

    left = 0;
    right = no-1;

    while (left <= right && !test)
    {
        mid = (int)((left+right)/2);
        if (x == list[mid]){
            test = 1;
            idx = mid;
        }
        else if (x>list[mid])
            left = mid+1;
        else
            right = mid-1;
    }
    return idx;
}
int main()
{
    int i, list[10];
    for(i=0;i<10;i++)
        list[i] = i+3*i;

    printf("\nOutput: %d ", testQ3(list,10,36));
    printf("\nOutput: %d ", testQ3(list,10,8));
    printf("\nOutput: %d ", testQ3(list,10,40));
    printf("\nOutput: %d ", testQ3(list,10,20));
    printf("\nOutput: %d ", testQ3(list,10,3));
    return 0;}

```

Figure Q3(b)

Answer:

(10 marks)

Q4 (a) Determine the results of each traversal algorithm for the tree in Figure Q4(a).

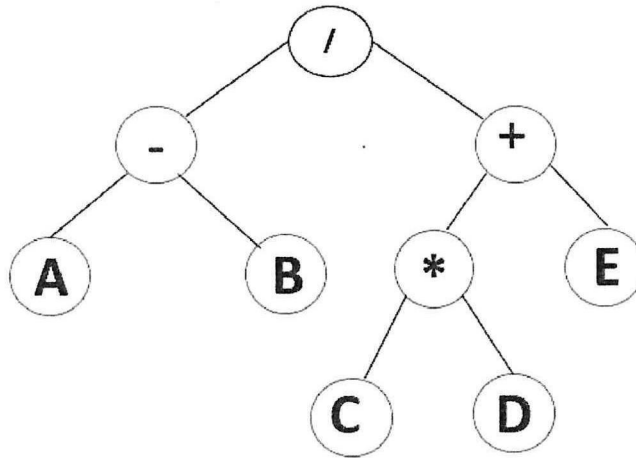


Figure Q4(a)

Traversal algorithm	Answer
Preorder	
Inorder	
Postorder	

(15 marks)

(b) Produce a binary search tree for the values in Figure Q4(b).

14, 10, 17, 12, 11, 20, 18, 25, 22, 23

Figure Q4(b)

Answer:

C3:10(10 marks)

- (c) Determine whether each of the following statements is **TRUE** or **FALSE** about graph.

Statements	Answer
Two standard ways of searching for a graph are depth-first search and breadth-first search. Both searching algorithms may begin at a designated vertex, but they differ with respect to the order in which they visit the remaining vertices.	
A graph may be represented using both, array and linked list.	
In a directed graph, some edges may not be assigned directions, therefore it is not the same as multigraph.	
A simple path is a path such that all vertices are distinct, except that the first and last could be the same.	
A graph is said to be complete if at least one node in the graph is adjacent to another node.	

(10 marks)

- END OF QUESTIONS -

TERBUKA