Universiti Tun Hussein Onn Malaysia

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FINAL EXAMINATION
SEMESTER II
SESSION 2021/2022

COURSE NAME	:	DATA STRUCTURE AND ALGORITHM
COURSE CODE	:	BIT 10703
PROGRAMME CODE	:	BIT
EXAMINATION DATE	:	JULY 2022
DURATION	:	3 HOURS
INSTRUCTIONS	:	1. ANSWER ALL QUESTIONS. 2. THIS EXAMINATION IS AN **ONLINE ASSESSMENT** AND CONDUCTED VIA **CLOSE BOOK.** 3. STUDENTS ARE **PROHIBITED** TO CONSULT THEIR OWN MATERIAL OR ANY EXTERNAL RESOURCES DURING THE EXAMINATIONS CONDUCTED VIA CLOSED BOOK.

THIS QUESTION PAPER CONSISTS OF **EIGHT (8)** PAGES

Q1 Answer **Q1(a)** and **Q1(b)** based on the information given in **Figure Q1**.

```
#include <stdio.h>
#include <stdlib.h>

struct Node{
      int data;
      struct Node *nextPtr;
};

typedef struct Node Node;
typedef Node* NodePtr;

void myFunction1(NodePtr *p1Ptr, NodePtr *p2Ptr);
void myFunction2(NodePtr *p1Ptr, NodePtr *p2Ptr, int value);
void myFunction3(NodePtr p1Ptr);
int myFunction4(NodePtr p1Ptr);
void myFunction5(NodePtr *p3Ptr, int info);
void myFunction6(NodePtr *p3Ptr);

int main()
{
      NodePtr p1Ptr = NULL, p2Ptr = NULL, p3Ptr = NULL;

      for (int i=2; i<12; i+=2)
      {
        myFunction2(&p1Ptr, &p2Ptr, i);
      }
      myFunction3(p1Ptr);
      myFunction1(&p1Ptr, &p2Ptr);
      myFunction1(&p1Ptr, &p2Ptr);
      myFunction3(p1Ptr);

      for (int i=10; i<20; i+=2)
      {
        myFunction5(&p3Ptr, i);
      }
      myFunction3(p3Ptr);
      myFunction6(&p3Ptr);
      myFunction6(&p3Ptr);
      myFunction3(p3Ptr);
      return 0;
}

void myFunction1(NodePtr *p1Ptr, NodePtr *p2Ptr)
{
      NodePtr myPtr;

      myPtr = *p1Ptr;
      *p1Ptr = (*p1Ptr)->nextPtr;

      if(*p1Ptr == NULL){
            *p2Ptr = NULL;
      }

      free(myPtr);
}
```

```
void myFunction2(NodePtr *p1Ptr, NodePtr *p2Ptr, int value)
{
      NodePtr myPtr;

      myPtr = malloc(sizeof(Node));

      if (myPtr!=NULL){
            myPtr->data = value;
            myPtr->nextPtr = NULL;

            if(myFunction4(*p1Ptr)){
                  *p1Ptr = myPtr;
            }
            else{
                  (*p2Ptr)->nextPtr = myPtr;
            }

            *p2Ptr = myPtr;
      }

}

void myFunction3(NodePtr p1Ptr)
{
      printf("The output is:\n");
            while(p1Ptr!=NULL){
                  printf("%d ",p1Ptr->data);
                  p1Ptr = p1Ptr->nextPtr;
            }
            printf(" Ending\n\n");
}

int myFunction4(NodePtr p1Ptr)
{ return (p1Ptr==NULL);}

void myFunction5(NodePtr *p3Ptr, int info)
{
      NodePtr myPtr;

      myPtr = malloc(sizeof(Node));

      if(myPtr!=NULL)
      {
            myPtr->data = info;
            myPtr->nextPtr = *p3Ptr;
            *p3Ptr = myPtr;
      }

}

void myFunction6(NodePtr *p3Ptr)
{
      NodePtr myPtr;

      myPtr = *p3Ptr;
      *p3Ptr = (*p3Ptr)->nextPtr;
      free(myPtr);
}
```

Figure Q1

(a) Write the output for the program in **Figure Q1**.

(14 marks)

(b) Determine whether each of the following statements is **TRUE** or **FALSE**.

(i) MyFunction1 demonstrates insertion algorithm for stack operation.
(2 marks)

(ii) MyFunction2 demonstrates insertion algorithm for queue operation.
(2 marks)

(iii) MyFunction3 returns NULL if a linked list is empty.
(2 marks)

(iv) MyFunction5 demonstrates insertion algorithm for stack operation.
(2 marks)

(v) MyFunction6 demonstrates insertion algorithm for queue operation.
(2 marks)

Q2 Answer **Q2(a)-Q2(c)** based on the information given in **Figure Q2**.

```
#include <stdio.h>
#define SIZE 10
int main()
{
	int myNumber1[SIZE]= {22,5,67,98,45,32,101,99,73,10};
	int myNumber2[SIZE]= {22,5,67,98,45,32,101,99,73,10};
	void myAlgo1(int n[],int size);
	void myAlgo2(int n[],int size);

	myAlgo1(myNumber1,SIZE);
	printf("\n\n");
	myAlgo2(myNumber2,SIZE);

	return 0;
}

void myAlgo1(int n[], int size)
{
	int i, j, min, minidx, temp;

	for(i=0; i<size-1; i++)
	{
		min = n[i];
		minidx = i;
		for(j=i+1; j<size; j++)
		{
```

```
                    if(n[j]<min)
                    {
                          min = n[j];
                          minidx = j;
                    }
              }

              if (min<n[i])
              {
                    temp = n[i];
                    n[i] = min;
                    n[minidx] = temp;
              }

              printf("\nSequence %d: ",i+1);
              for(int k=0; k<size; k++)
                 printf("%d   ",n[k]);
        }
}

void myAlgo2(int n[],int size)
{
      int i, j, temp;

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

            printf("\nSequence %d: ",i+1);
            for(int k=0; k<size; k++)
               printf("%d   ",n[k]);
      }
}
```

Figure Q2

(a) Write the output for the program.

(20 marks)

(b) Name the algorithm for `Algo1`.

(2 marks)

(c) Name the algorithm for `Algo2`.

(2 marks)

Q3 Answer **Q3(a)-Q3(c)** based on the information given in **Figure Q3**.

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
#define SIZE 10

int main()
{
      int n1[SIZE]= {35,40,52,62,75,97,103,128,129,131};
      int n2[SIZE]= {235,40,152,62,175,971,10,12,129,131};
      int item, myLocation;
      int mySearch1(int n[],int size,int item);

      printf("\nEnter the item you are searching for: ");
      scanf("%d", &item);
      myLocation = mySearch1(n1,SIZE,item);

      if(myLocation>-1)
         printf("Item is found at index %d.",myLocation);
      else
         printf("Item is not found in the array.");

      return 0;
}
```

Figure Q3

(a) Write the `mySearch` function. The `mySearch` function is applicable to find an item from an array with sorted order values only, as declared in **Figure Q3**.

(20 marks)

(b) Name an algorithm that is applicable to search an item from the array, called `n2` declared in **Figure Q3**.

(2 marks)

Q4 (a) Determine whether each of the following statements is **TRUE** or **FALSE** about tree and graph.

(i) A tree is a particular type of a graph.

(1 mark)

(ii) A graph is a data structure that includes nodes similar to what we used in creating linked lists.

(1 mark)

(iii) In a graph, nodes can also be called edges.

(1 mark)

(iv) Two nodes are considered adjacent when no edge connects them.

(1 mark)

(v) A simple path exists in a graph whenever that path has no repeated vertices.

(1 mark)

(vi) A real life application of graph is in simulating a map with the edges being cities and the edges being roads between the cities.

(1 mark)

(vii) A weighted graph can be created to model distance between cities.

(1 mark)

(viii) In a binary tree, only one node is allowed to have more than two children.

(1 mark)

(ix) We can use stack to create a binary tree.

(1 mark)

(x) A tree node can have only two children.

(1 mark)

(xi) A binary tree node may have node without children.

(1 mark)

(xii) A graph is a special instance of a tree.

(1 mark)

(xiii) The root of a tree typically is drawn at the bottom of the tree.

(1 mark)

(xiv) A tree structure may have two tree pointer variables as members.

(1 mark)

(xv) A node that has no children is called a leaf.

(1 mark)

(b) Based on the information given in **Figure Q4,** write the results for each of the following traversal algorithms.

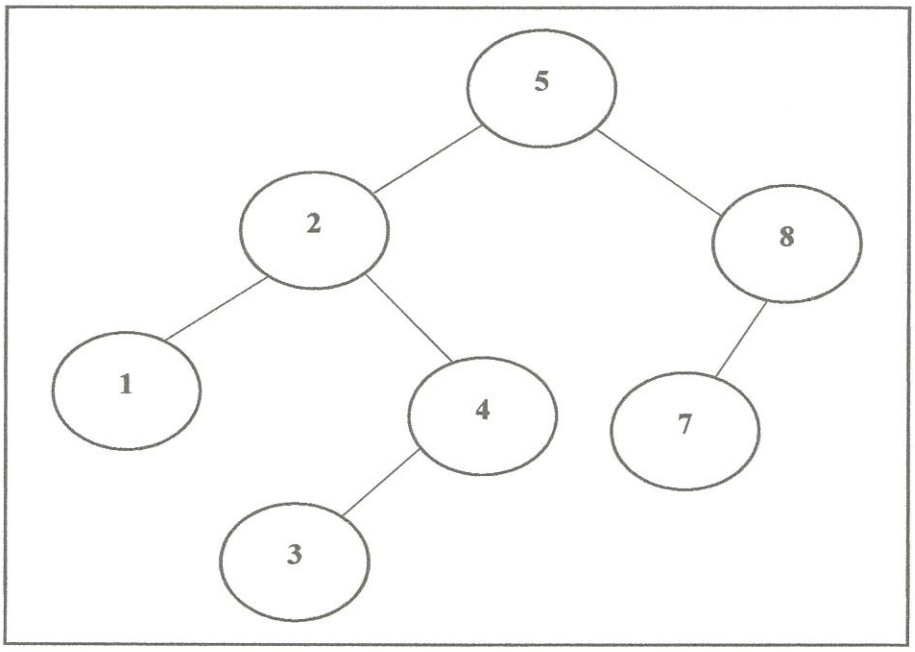

Figure Q4

(i) Preorder

(5 marks)

(ii) Inorder

(5 marks)

(iii) Postorder

(5 marks)

- END OF QUESTIONS -