Universiti Tun Hussein Onn Malaysia

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FINAL EXAMINATION
SEMESTER II
SESSION 2023/2024

COURSE NAME	:	DATA STRUCTURE AND ALGORITHM
COURSE CODE	:	BIT 10703
PROGRAMME CODE	:	BIT
EXAMINATION DATE	:	JULY 2024
DURATION	:	3 HOURS
INSTRUCTIONS	:	1. ANSWER ALL QUESTIONS 2. THIS FINAL EXAMINATION IS CONDUCTED VIA ☐ Open book ☒ Closed book 3. STUDENTS ARE **PROHIBITED** TO CONSULT THEIR OWN MATERIAL OR ANY EXTERNAL RESOURCES DURING THE EXAMINATION CONDUCTED VIA CLOSED BOOK

THIS QUESTION PAPER CONSISTS OF **ELEVEN (11)** PAGES

Q1 Answer **Q1(a)-Q1(b)**, according to the declaration of `arrayNumber` given as follows:

```
int arrayNumber[11] = {78, 65, 101, 59, 11, 25, 82, 39, 15, 47, 50};
```

(a) Show the content of the array after every pass of bubble sort algorithm.

(10 marks)

Answer:

(b) Show the content of the array after every pass of selection sort algorithm.

(10 marks)

Answer:

Q2 Determine the output for the code given in **Figure Q2.1**.

```
#include <stdio.h>

int callMe(int n1, int n2);
int main()
{
      int num1=4, num2=4;
      printf("%d ", callMe(num1, num2));
}

int callMe(int n1, int n2)
{
      if (n2==0)
      {
            printf("\nCall One.");
            return 1;
      }

      else
      {
            printf("\nCall Two.");
            return n1*callMe(n1,n2-1);
      }

}
```

Figure Q2.1

(7 marks)

Answer:

Q3 Answer **Q3(a)-Q3(b)** based on the code given in **Figure Q3.1**.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXCHARS 30
#define SIZE 7

struct CovidCase{
      char areaName[MAXCHARS];
      int totalPatients;
};

typedef struct CovidCase CovidCase;

struct Node{
      CovidCase data;
      struct Node *next;
};

typedef struct Node Node;
typedef Node* NodePtr;

void myFunction1(NodePtr *topPtr, CovidCase data);
CovidCase myFunction2(NodePtr *topPtr);
void myFunction3(NodePtr *ptr1, NodePtr *ptr2, CovidCase data);
CovidCase myFunction4(NodePtr *ptr1, NodePtr *ptr2);
int myFunction5(NodePtr topPtr);
void myFunction6(NodePtr currPtr);

int main()
{
      NodePtr myPtr1 = NULL, myPtr2 = NULL, myPtr3 = NULL;

      CovidCase myCovidCase;
      CovidCase covidCaseArray[SIZE]={{"Johor Bahru",8500},
                                      {"Kulai",8000},
                                      {"Kluang",780},
                                      {"Kota Tinggi",958},
                                      {"Batu Pahat",9000},
                                      {"Muar",6000},
                                      {"Pontian",1222}};
```

```
	for(int idx=0; idx<SIZE; idx++)
	{
		myFunction1(&myPtr1,covidCaseArray[idx]);
		myFunction3(&myPtr2,&myPtr3,covidCaseArray[idx]);
	}

	myCovidCase = myFunction2(&myPtr1);
	printf("\nFocus Area 1 %s  %d",myCovidCase.areaName,
					myCovidCase.totalPatients);
	printf("\n");
	printf("\nCovid Cases 1 :\n");
	myFunction6(myPtr1);
	myCovidCase = myFunction4(&myPtr2,&myPtr3);
	printf("\nFocus Area 2 %s  %d",myCovidCase.areaName,
					myCovidCase.totalPatients);
	printf("\n");
	printf("\nCovid Cases 2 :\n");
	myFunction6(myPtr2);
}

void myFunction1(NodePtr *testPtr, CovidCase data)
{
	NodePtr newPtr;

	newPtr = malloc(sizeof(Node));

	if(newPtr!=NULL)
	{
		newPtr->data = data;
		newPtr->next = *testPtr;
		*testPtr = newPtr;
	}
	else
	{
		printf("\nComplete MyFunction1.");
	}
}

CovidCase myFunction2(NodePtr *topPtr)
{
	NodePtr tempPtr;
	CovidCase testData;

	tempPtr = *topPtr;
	testData = (*topPtr)->data;
	*topPtr = (*topPtr)->next;
	free(tempPtr);

	return testData;
}

void myFunction3(NodePtr *ptr1, NodePtr *ptr2, CovidCase data)
{
	NodePtr myNode;

	myNode = malloc(sizeof(Node));

	if(myNode!=NULL)
	{
		myNode->data = data;
```

```
            myNode->next =NULL;

            if(myFunction5(*ptr1))
            { *ptr1 = myNode;}
            else
            {(*ptr2)->next = myNode;}

            *ptr2 = myNode;
      }
      else
          printf("\nComplete MyFunction3.");
}


CovidCase myFunction4(NodePtr *ptr1, NodePtr *ptr2)
{
      CovidCase myData;
      NodePtr myNode;

      myData = (*ptr1)->data;
      myNode = *ptr1;
      *ptr1 = (*ptr1)->next;

      if(*ptr1==NULL)
        *ptr2 = NULL;

      free(myNode);
      return myData;
}

int myFunction5(NodePtr testPtr)
{
      return testPtr == NULL;
}

void myFunction6(NodePtr ptr)
{
      if (ptr==NULL)
      {
            printf("No Output.\n\n");
      }
      else
      {
            while(ptr!=NULL)
            {
                  printf("%s %d\n",ptr->data.areaName,
                                    ptr->data.totalPatients);
                  ptr = ptr->next;
            }

            printf("\n");
      }
}
```

Figure Q3.1

(a) Determine the output for the code in **Figure Q3.1**.

(16 marks)

Answer:

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

(10 marks)

No.	Statement(s)	Answer
(i)	`myFunction1()` and `myFunction2()` are applicable to insert data into a linked list.	
(ii)	`myFunction1()` and `myFunction3()` are applicable to insert data to a linked list.	
(iii)	`myFunction2()` and `myFunction4()` are applicable to demonstrate the concept of First-In-First-Out using array.	
(iv)	`myFunction1()` and `myFunction2()` are applicable to demonstrate the concept of stack using array.	
(v)	`myFunction3()` and `myFunction4()` are applicable to demonstrate the concept of queue using linked list.	
(vi)	`myFunction1()` and `myFunction2()` are applicable to demonstrate the concept of Last-In-First-Out .	
(vii)	`myFunction1()` applies the steps of push operation for a stack.	
(viii)	`myFunction1()` applies the steps of dequeue operation for a queue.	
(ix)	`myFunction2()` applies the steps of dequeue operation for a queue.	
(x)	`myFunction5()` applies the steps of pop operation for a stack.	

Q4 Azza Tuition Centre would like to identify lecturers who have at least 40 students to consider for monthly bonus. Given the code in **Figure Q4.1**, write a program fragment that will list the lecturers who are eligible for the bonus with their corresponding total students.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXCHARS 30
#define SIZE 5

struct Lecturer{
      char name[MAXCHARS];
      int totalStudents;
};

typedef struct Lecturer Lecturer;

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

typedef struct Node Node;
typedef Node* NodePtr;

void insert(NodePtr *sPtr, Lecturer data);

int main()
{
      NodePtr startPtr = NULL;
      char inputName[MAXCHARS];
      int inputTotal;
      Lecturer inputData;

      for(int idx=0; idx<SIZE; idx++)
      {
            printf("\nEnter lecturer's name: ");
            scanf("%s", &inputName);
            printf("\nEnter total students: ");
            scanf("%d", &inputTotal);
            strcpy(inputData.name,inputName);
            inputData.totalStudents = inputTotal;
            insert(&startPtr,inputData);
      }
      return 0;
}

void insert(NodePtr *sPtr, Lecturer data)
{
      NodePtr newPtr;
      NodePtr prevPtr;
      NodePtr currPtr;

      newPtr = malloc(sizeof(Node));

      if(newPtr!=NULL){
            newPtr->data = data;
            newPtr->nextPtr = NULL;
```

```
            prevPtr = NULL;
            currPtr = *sPtr;

while(currPtr!=NULL&&data.totalStudents>currPtr>data.totalStudents)
            {
                  prevPtr = currPtr;
                  currPtr = currPtr->nextPtr;
            }

            if (prevPtr == NULL){
                  newPtr->nextPtr = *sPtr;
                  *sPtr = newPtr;
            }
            else{
                  prevPtr->nextPtr = newPtr;
                  newPtr->nextPtr = currPtr;
            }

      }
      else{
            printf("Insertion error.");
      }
}
```

Figure Q4.1

(9 marks)

Answer:

Q5 Answer **Q5(a)-Q5(b)** based on the tree given in **Figure Q5.1**.

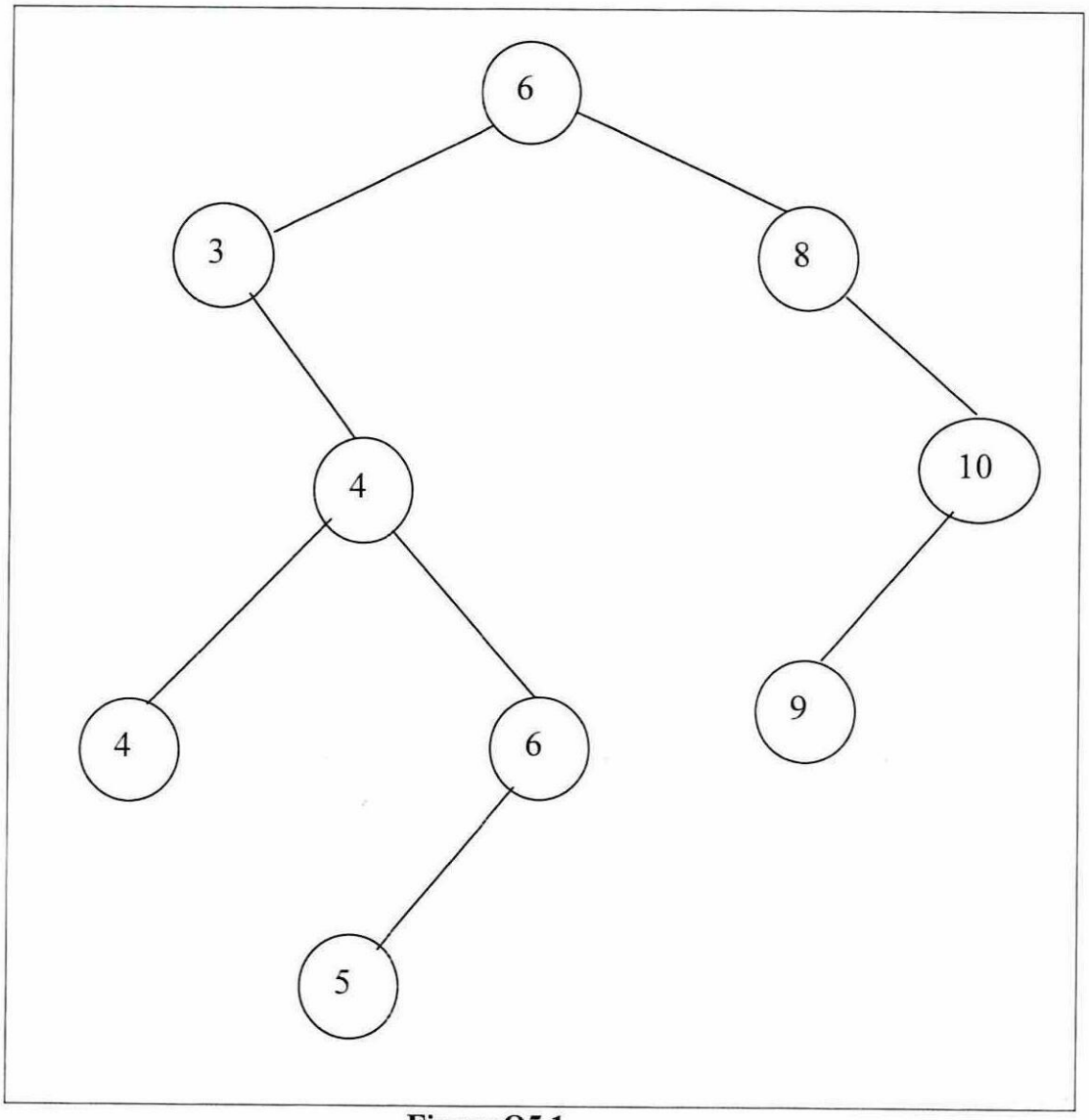

Figure Q5.1

(a) Write the sequence of the values after performing preorder traversal algorithm.

(9 marks)

Answer:

(b) Write the sequence of the values after performing postorder traversal algorithm.

(9 marks)

Answer:

- END OF QUESTIONS -