**TEST I: CP 123 DATA STRUCTURES AND ALGORITHIMS**

**PART 1: Structures**

**Question No.1**

1. Consider any object of your choice (eg: student, prisoner, car, drug, bag, shoes, etc) define a data structure pertinent to such object [5 Marks]
2. Modify the structures at a such that one of the fields is a pointer to such structure [3 Marks]
3. Write a code to implement a ***linked list*** of at least five [5] nodes of the structures defined in b [7 Mark]

**Question No.2**

1. Write a code to implement a queue of at least five [5] nodes of the structure defined in b [75 Marks]
2. Write a code to dequeua node from your queue [5 Marks]

**Question No.3**

1. Write a code to implement a stock of at least five [5] nodes of the structures defined in b [75 Marks]
2. Write a code to push() data in your stack [5 Marks]
3. Write a code to pop() data from your stack

**Question No.4**

1. What is the difference between linked lists and queues?(5)
2. Write a code to implement a binary tree of just to levels of braches(10)

**SOLUTIONS**

**Question No.1**

1. **.**

**#include <iostream>**

**Using namespace std;**

**//constructing Structure**

**Struct Student {**

**String name;**

**Int marks;**

**char grade; };**

**//Entering the main function**

**int main() {**

**//declaring an object of data type results**

**Student s1;**

**//assignment of values to objects1**

**s1.name="MUHINA"; s1.marks=100; s1.grade='A';**

**//displaying output on the screen**

**cout<<"The following are the marks for our students"<<"\n\n";**

**cout<<"Name:"<<s1.name<<"\n\n"; cout<<"Marks:"<<s1.marks<<"\n\n"; cout<<"Grade:"<<s1.grade;**

**return 0;**

**}**

**(b).**

**#include <iostream>**

**Using namespace std;**

**//constructing Structure**

**Struct Student {**

**String name;**

**Int marks;**

**char grade; };**

**//Entering the main function**

**int main() {**

**//declaring an object of data type results**

**Student s1;**

**Student \*s1, 1;**

**s1 = &1;**

**//assignment of values to objects1**

**s1.name="MUHINA"; s1.marks=100; s1.grade='A';**

**//displaying output on the screen**

**cout<<"The following are the marks for our students"<<"\n\n";**

**cout<<”Name:”<<(\*s1).name<<"\n\n";**

**cout<<"Marks:"<< (\*s1).marks<<"\n\n";**

**cout<<"Grade:"<< (\*s1).grade;**

**return 0;**

**}**

**(c).**

**#include <iostream>**

**Using namespace std;**

**//constructing Structure**

**struct student {**

**String name;**

**Int marks;**

**struct student \*grade; };**

**//create displaying function**

**void displaydata (student \*n) {**

**while ( n! = 0) {**

**cout << n -> name <<endl;**

**cout << n-> marks <<endl;**

**n = n -> grade; }**

**}**

**//Entering the main function**

**int main() {**

**//create empty node**

**student \*first = 0;**

**student \*second = 0;**

**student \*third = 0;**

**student \*fourth = 0;**

**student \*last = 0;**

**//put node on a heap**

**first = new student();**

**second = new student();**

**third = new student();**

**fourth = new student();**

**last = new student();**

**//Insert data**

**first -> name = “Madaraka”;**

**first -> marks = 90;**

**first -> grade = second;**

**second -> name = “Mboma”;**

**second -> marks = 70;**

**second -> grade = third;**

**third -> name = “Ngosha”;**

**third -> marks = 90;**

**third -> grade = fourth;**

**fourth -> name = “Cheta”;**

**fourth -> marks = 50;**

**fourth -> grade = last;**

**last -> name = “Mwakitalu”;**

**last -> marks = 60;**

**last -> grade = 0; NULL;**

**displaydata (first);**

**return 0;**

**}**

**Question No.2**

**#include <iostream>**

**using namespace std;**

**//declare structure in global**

**// on the nmodification of the first question**

**struct Student\_details**

**{**

**int reg;**

**//string name;**

**string gender;**

**// lets change the name from the string name to pointer**

**Student\_details \* name;**

**};**

**// decleration of the node/pointers**

**Student\_details \* front , \* endlink;**

**//creating a function to be able to create a linked list of queue**

**void create(int data, string sgender)**

**{**

**//create a new node**

**Student\_details \* newStudent = new Student\_details();**

**newStudent->reg = data;**

**newStudent->gender =sgender ;**

**newStudent->name =NULL;**

**if(front ==NULL && endlink ==NULL)**

**{**

**front = endlink = newStudent;**

**}**

**else{**

**endlink->name = newStudent;**

**endlink = newStudent;**

**}**

**}**

**//create a function to display data in the queue**

**void display()**

**{**

**Student\_details \* point;**

**if(front == NULL && endlink ==NULL)**

**{**

**cout << " queue is empty";**

**}**

**else**

**{**

**point = front;**

**while(point != NULL)**

**{**

**cout << point->reg <<endl;**

**cout << point ->gender <<endl;**

**point = point->name;**

**}**

**}**

**}**

**int main()**

**{**

**int value;**

**string gender;**

**cout <<"Enter student reg no" ;cin >>value ;**

**cout << "Enter student gender "; cin >> gender;**

**create(value,gender);**

**cout <<"Enter student reg no" ;cin >>value ;**

**cout << "Enter student gender "; cin >> gender;**

**create(value,gender);**

**cout <<"Enter student reg no" ;cin >>value ;**

**cout << "Enter student gender "; cin >> gender;**

**create(value,gender);**

**display();**

**// cout << " entered student details "<< endl;**

**//while(Student !=0 )**

**// {**

**// cout << " Student Reg number : " << Student->reg; cout <<endl;**

**// cout << " Student Gender : " << Student->gender; cout <<endl;**

**//**

**//Student = Student->name;**

**//}**

**}**

**Question No.3**

**#include <bits/stdc++.h>**

**using namespace std;**

**// A structure to represent a stack**

**//Make sure you change the name to your defined structure**

**struct StackNode {**

**int data;**

**StackNode\* next;**

**};**

**//3(a)Define stack**

**StackNode\* newNode(int data)**

**{**

**StackNode\* stackNode = new StackNode();**

**stackNode->data = data;**

**stackNode->next = NULL;**

**return stackNode;**

**}**

**//(3b)Pushing onto stack**

**int isEmpty(StackNode\* root)**

**{**

**return !root;**

**}**

**void push(StackNode\*\* root, int data)**

**{**

**StackNode\* stackNode = newNode(data);**

**stackNode->next = \*root;**

**\*root = stackNode;**

**cout << data << " pushed to stack\n";**

**}**

**//3(c)**

**int pop(StackNode\*\* root)**

**{**

**if (isEmpty(\*root))**

**return INT\_MIN;**

**StackNode\* temp = \*root;**

**\*root = (\*root)->next;**

**int popped = temp->data;**

**free(temp);**

**return popped;**

**}**

**int peek(StackNode\* root)**

**{**

**if (isEmpty(root))**

**return INT\_MIN;**

**return root->data;**

**}**

**// Driver code**

**int main()**

**{**

**StackNode\* root = NULL;**

**push(&root, 10);**

**push(&root, 20);**

**push(&root, 30);**

**cout << pop(&root) << " popped from stack\n";**

**cout << "Top element is " << peek(root) << endl;**

**cout<<"Elements present in stack : ";**

**//print all elements in stack :**

**while(!isEmpty(root))**

**{**

**// print top element in stack**

**cout<<peek(root)<<" ";**

**// remove top element in stack**

**pop(&root);**

**}**

**return 0;**

**}**

**Question No.4**

1. **.**

|  |  |
| --- | --- |
| **QUEUES** | **LINKED LISTS** |
| **Queue** is a **"linear data structure"**that works on the **principle** of **"first in first out"**(**FIFO**). | **Linked list** is also a form of **"linear data structure"**, where in the data is stored along with the **memory address**to the next data item |
| In queue least recently added item is removed first | data can be **added or deleted randomly** without maintaining any sequence. |
| whereas **queue** follows **first come first serve principle.** | a data item can be appended to first element or last or any of the other middle elements in a linked list, |
| Static queue is fixed in size and could not be altered. The size of queue is defined at the time of its declaration. | List does not have fixed size and so no size is required at the time of declaration of list. |
| internally queue has implemented array which makes it faster for searching and addition of elements. | On the other hand, list maintains node and pointers for storing data and address of the next node |
| The static queue is always based on First in first out (FIFO) technique | On the other hand List can be FIFO or Last in First out (LIFO) |

**NOTE: Not All of these Above Solutions are VALID SOLNS**

**&**

**Any Variable may changes according to your choice**