Thapar Institute of Engineering & Technology, Patiala

Department of Computer Science and Engineering

MID SEMESTER EXAMINATION (Solution)

B. E. (First Year): Semester-II (2019/20) (CSBS)	Course Code: UCT203 Course Name: Data Structures and Algorithms
05 March, 2019	Time: 1:00 P.M. - 3:00 P.M.
M. Marks: 30	Name of Faculty: Dr. Rajendra Ku. Roul

Q1. Write a function in C which will insert integers to a circular array. Your code should have provision to check the *overflow condition* suitably. [**4]**

Solution:

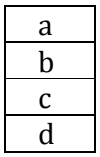
```
#include <stdio.h>
int a[5], f = 0, r = 0, flag = 0;
int main ( )
{
 void push ( );
 return 0;
}
void push ( )
{
 int n;
 if ((f == r) && (flag == 1))
  printf ("overflow");
 else
  {
   printf ("\n enter the element");
   scanf ("%d", &n);
   a[r] = n;
   r++;
   if (r == 5)
        {
         r = 0;
         flag = 1;
        }
  }
}
```

Q2. Write a recursive function to print all the elements of a single linked list in a reverse order. [**4]**

Solution:

```
void Linklist(struct node *head)
{
        if(head==NULL)
        return;
        else
        {
                Linklist(head->next);
                printf("%d", head->item);
        }
}
```

Q3. Stack **A** has the entries a, b, c, d (with **a** on the top). An entry popped out of stack **A** can be printed immediately or pushed to stack **B**. An entry popped out of stack **B** can only be printed. In this arrangement, how many permutations of a, b, c, d which are *not* possible and list them. [**3]**



A

Solution:

There are 10 such possible cases which are not possible. They are as follows:

a d b c, b d a c, c a b d, c a d b, c d a b, d a b c, d a c b, d b a c, d b c a, d c a b

Q4. a) Discuss different dynamic memory allocation functions used in C programming. What is the main difference between malloc( ) and calloc( )? [**4+2]**

Solution:

i)malloc( ): "malloc" or **"memory allocation"** method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

Syntax: ptr=(cast-type*)malloc(byte-size)

ii)calloc( ):"calloc" or **"contiguous allocation"** method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value '0'.

Syntax: ptr=(cast-type*)calloc(number, byte-size)

iii)realloc( ):"realloc" or **"re-allocation"** method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory

previously allocated with the help of **malloc( )** or **calloc( )** is insufficient, **realloc( )** can be used to **dynamically re-allocate memory**.

Syntax: ptr=realloc(ptr, new-size)

iv) free( ):"free" method in C is used to dynamically **de-allocate** the memory. The memory allocated using functions **malloc()** and **calloc( )** is not de-allocated on their own. Hence the **free( )** method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

Syntax: free(ptr)

There are two major differences between **malloc( )** and **calloc( )** in C programming language: first, in the number of arguments. The **malloc**() takes a single argument, while **calloc**() takess two. Second, **malloc**() does not initialize the memory allocated, while **calloc**() initializes the allocated memory to 0.

b) Discuss different mathematical notations used to measure the time complexity of an algorithm. What is best case and worst case time complexity**?** **[3+2]**

Solution:

i)Big-O notation:

O(g(n)) = { f(n): there exist positive constants c and n0 such that 0 <= f(n) <= c*g(n) for all n >= n0}

ii) Big-Omega notation:

Ω (g(n)) = {f(n): there exist positive constants c and n0 such that 0 <= c*g(n) <= f(n) for all n >= n0}.

iii) Big-theta notation:

Θ(g(n)) = {f(n): there exist positive constants c1, c2 and n0 such that 0 <= c1*g(n) <= f(n) <= c2*g(n) for all n >= n0}

- **Best case** = fastest time to complete, with optimal inputs chosen. For example, the best case for a sorting algorithm would be data that's already sorted.
- **Worst case** = slowest time to complete, with pessimal inputs chosen. For example, the worst case for a sorting algorithm might be data that's sorted in reverse order (but it depends on the particular algorithm).

Q5. Write a function in C to delete the first node of a double linked list. **[5]**

Solution:

```
Delete_firstnode(struct node *head)
{
        struct node *p=head;
        head=head->next;
        head->prev=NULL;
        free(p);
}
```

Q6. Assume the structure of a linked list as follows:

```
struct Node
{
  int data;
  struct Node *next;
};
```

What does the following function do for a given Linked List? **[3]**

```
void fun2(struct Node* head)
{
  if(head== NULL)
    return;
  printf("%d", head->data);
    if(head->next != NULL )
    fun2(head->next->next);
  printf("%d", head->data);
 }
```

Solution:

fun2( ) prints alternate nodes of the given Linked List, first from head to end, and then from end to head. If Linked List has even number of nodes, then fun2( ) skips the last node. For Linked List 1->2->3->4->5, fun2( ) prints 1 3 5 5 3 1.

For Linked List 1->2->3->4->5->6, fun2( ) prints 1 3 5 5 3 1.