Tutorial

Programming & Data Structure: CS 11001

Section - 4/D

Department of Computer Science and Engineering

I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (20.03.2014)

Download

Download the file tut200314.pdf from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

Dynamic Allocation of Memory

```
int *addr(){      // heapStack1.c
    int n, *p;

    n = 10;
    p = (int *)malloc(sizeof(int));
    printf("&n: %p, p: %p\n", &n,  p);
    *p = 10;
    return p;
}
```

Dynamic Allocation of Memory

Try with `heapStack2.c`

Dynamic Allocation of 1D Array

```
int main()  // 1dDyArray.c
{
    int *p, n, i;
    scanf("%d", &n);
    p = (int *)malloc(n*sizeof(int));
    for(i=0; i<n; ++i) p[i] = i;
    for(i=0; i<n; ++i) printf("%d ", p[i]);
    putchar('\n'); return 0;
}
```

Tutorial X.1

Write a C function `int *createArray(int n)` that takes a positive integer `n` as parameter, creates a dynamic array of size n integers, initializes locations with 0, and returns the starting address.

A Simple Self-Referencing Structure

```
typedef struct node {
        int data ;
        struct node *next ;
} node, *list ;
```

Tutorial X.2

What will be printed by the following code?

```
int main()
{
    node a; list l;
    printf("%d %d\n", sizeof a, sizeof l);
    return 0;
}
```

Tutorial X.3

What will be printed by the following code?

```
int main()
{
    node a = {5, &a}; list l = a.next;
    printf("%d\n", a.data*(*l).data+l->data);
    return 0;
}
```

Tutorial X.4

How do yo dynamically create an object of type `node` that will be pointed by `l`, and put a data in the node.

```
int main() {
    list l ;

    l =
    return 0;
}
```

Tutorial X.5

Following code in `main()` reads `n` data an insert them at the head of a list pointed by `l` (originally NULL). Fill-up the missing code.

```
list l = NULL;
printf("Enter %d data:\n", n);
for(i=1; i<=n; ++i) {
    list l1;
    l1 =                    ;
    scanf("%d",            );
                            ;
                            ;
}
```

Tutorial X.6

What do you 'expect' the following `insert()` function to do?

```
void insert(list l, int data){
     list tP = (list)malloc(sizeof(node));
     tP->next = l;
     tP->data = data;
     l = tP;
}
```

Is there anything wrong?

Tutorial X.7

What are the different ways you modify the previous function so that it works as 'expected'?

Tutorial X.8

Write a recursive function to print the data present in the list starting from its head.

```
void printList(list l)
```

Tutorial X.9

Write a recursive function to print the data present in the list starting from its tail towards the head.

```
void printList(list l)
```

Tutorial X.10

Write a recursive function to reverse a list (no new list is created).

```
list reverseList(list l)
```