Sorting a singly linked list

```
#include<iostream>
#include<stdlib.h>
using namespace std;

/* List Structure */
typedef struct Node
{
        int data;
        struct Node *link;
}node;

node *head = NULL;                      // Head node to keep track of linked list

/* Driver functions */
void print();
void swap(node *p1, node*p2);
void SelectionSort(node *head);
void insert(int data, int position);


/* Main method */
int main()
{
        insert(4,1);    // Insert Element at first position LINKED-LIST =  / 4 /
        insert(2,2);    // Insert Element at second position LINKED-LIST = / 4 2 /
        insert(3,3);    // Insert Element at third position LINKED-LIST =  / 4 2 3 /
        insert(1,4);    // Insert Element at fourth position LINKED-LIST = / 4 2 3 1/
        insert(0,5);    // Insert Element at fifth position LINKED-LIST =  / 4 2 3 1 0/

        printf("\n Before sorting = ");
        print();

        SelectionSort(head);            //  Sorting linked list

        printf("\n After sorting  = ");
        print();

        return 0;
}
```

```
/* To sort the linked list */
void SelectionSort(node *head)
{
	node *start = head;
	node *traverse;
	node *min;

	while(start->link)
	{
		min = start;
		traverse = start->link;

		while(traverse)
		{
			/* Find minimum element from array */
			if( min->data > traverse->data )
			{
				min = traverse;
			}

			traverse = traverse->link;
		}
		swap(start,min);				// Put minimum element on starting location
		start = start->link;
	}
}

/* swap data field of linked list */
void swap(node *p1, node*p2)
{
	int temp = p1->data;
	p1->data = p2->data;
	p2->data = temp;
}

/* Function for Inserting nodes at defined position */
void insert(int data, int position)
{
	/* Declaring node */
	node *temp = (node*)malloc(sizeof(node));
	temp->data = data;
	temp->link = NULL;
```

```
        /* if node insertion at first point */
        if(position==1)
        {
        temp->link = head;
        head = temp;
        return ;
        }

        /* Adding & Adjusting node links*/
        node *traverse = head;
        for(int i=0; i<position-2; i++)
        {
        traverse = traverse->link;
        }
        temp->link = traverse->link;
        traverse->link = temp;
}

/* Function for Printing Linked List */
void print()
{
        node *p = head;

        while(p)
        {
                printf(" %d",p->data);
                p = p->link;
        }
        printf(" \n\n");
}
```