Question 1

```
#include <stdio.h>

#include <stdlib.h>


typedef struct Node

{

   int data;

   struct Node *next;

}node;


node *create(int x)

{

   node *new=(node*)malloc(sizeof(node));

   new->data=x;

   new->next=NULL;


   return new;

}


node* insert_at_beginning(node* head,int x)

{

   node* new=create(x);

   new->next=head;

   head=new;

   return head;

}
```

```
node *insert_any(node* head,int x, int val)
{
    if (val == 1)
    {
        return insert_at_beginning(head,x);

    }

    node *new=create(x);
    node* temp=head;



    if(!head)
    {
        head=new;
        return head;
    }
    for (int i=0;i<val-2;i++)
    {
       temp=temp->next;
    }

    node *replace = temp->next;

    temp->next = new;
    new->next = replace;

    return head;
}
```

```
int main()
{
    struct Node *head = NULL;
    struct Node *second = NULL;
    struct Node *third = NULL;

    // allocate 3 nodes in the heap
    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));

    head->data = 1;      // assign data in first node
    head->next = second; // Link first node with second

    second->data = 2; // assign data to second node
    second->next = third;

    third->data = 3; // assign data to third node
    third->next = NULL;

    int position,value;
    printf("Enter a number of position: ");
    scanf("%d", position);

    printf("Enter a value: ");
    scanf("%d", value);
```

```
        head=insert_any(head,value,position);


        return 0;
    }
```

Question 2

```
node *delete_beg(node *head)
{
    if(!head)
      return head;
    node *temp = head;
    temp = temp->next;
    free(head);
    head = temp;
    return temp;
}
```

Question 3

```
node* delete_end(node* head)
{
    if(!head || !head->next)
    {
        free(head);
        return NULL;
    }
    node* temp=head;
    node* previous=NULL;
    while(temp->next)
    {
        previous=temp;
        temp=temp->next;
    }
    free(previous->next);
    previous->next=NULL;
    return head;
}
```