

Name: KUSUM.M.R  
VSN: 1BM19CS077  
LAB PROGRAM - 8

09/12/2020

WAP to implement Stack and Queue using Linked Representation

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int info;
    struct node *link;
};

typedef struct node *NODE;
```

```
NODE getnode()
```

```
{
```

```
    NODE x;
```

```
    x = (NODE) malloc (sizeof (struct node));
```

```
    if (x == NULL)
```

```
    {
        printf ("Memory full \n");
        exit (0);
    }
```

```
    return x;
```

```
}
```

```
void freenode (NODE x)
```

```
{
```

```
    free(x);
```

```
}
```

```
NODE insert_node_front (NODE first, int item)
```

```
{
```

```
    NODE temp;
```

```
    temp = getnode();
```

```
    temp → info = item;
```

```
    temp → link = NULL;
```

```
    if (first == NULL)
```

```
        return temp;
```

```
    temp → link = first;
```

```
    first = temp;
```

```
    return first;
```

```
}
```

```
NODE insert_rear (NODE first, int item)
```

```
{
```

```
    NODE temp, cur;
```

```
    temp = getnode();
```

```
    temp → info = item;
```

```
    temp → link = NULL;
```

```
    if (first == NULL)
```

```
        return temp;
```

```
    cur = first;
```



```

while (cur → link != NULL)
    cur = cur → link;
cur → link = temp;
return first;
}

```

NODE delete-front (NODE first)

```

{
    NODE temp;
    if (first == NULL)
    {
        printf("List is empty cannot delete\n");
        return first;
    }
    temp = first;
    temp = temp → link;
    printf("Item deleted at front-end is = %d\n", first → info);
    free(first);
    return temp;
}

```

NODE delete-rear (NODE first)

```

{
    NODE cur, prev;
    if (first == NULL)
    {
        printf("List is empty cannot delete\n");
        return first;
    }
    if (first → link == NULL)
    {
        printf("Item deleted is %d\n", first → info);
        free(first);
        return NULL;
    }
    prev = NULL;
    cur = first;
}

```

```
while (cur → link != NULL)
```

```
{  
    prev = cur;  
    cur = cur → link;
```

```
    printf("Item deleted at rear end is %d", cur → info);  
    free(cur);  
    prev → link = NULL;  
    return first;
```

```
}
```

```
void display(NODE first)
```

```
{  
    NODE temp;
```

```
    if (first == NULL)
```

```
{  
        printf("List is empty cannot display items \n");  
        return;
```

```
}
```

```
    printf("Contents of list : \n");
```

```
    for (temp = first; temp != NULL; temp = temp → link)
```

```
{  
        printf("%d \n", temp → info);
```

```
}
```

```
}
```

```
void main()
```

```
{  
    int item, choice, pos, i, n, count, key;
```

```
    NODE first = NULL, a, b;
```

```
    for (;;) 
```

```
{  
        printf("\n 1: Stack \n 2: Queue \n 3: Exit \n");
```

```
        printf("Enter the choice \n");
```

```
        scanf("%d", &choice);
```

```
        switch (choice)
```

```
{
```



```
case 1: printf("stack\n");  
for (; ;)
```

```
{ printf("\n1: Insert-rear\n2: Delete-rear\n3: Display-  
list\n4: Exit\n");  
printf("Enter the choice\n");  
scanf("%d", &choice);  
switch (choice)
```

```
{  
case 1: printf("Enter the item at rear-end\n");  
scanf("%d", &item);  
first = insert-rear (first, item);  
break;  
case 2: first = delete-rear (first);  
break;  
case 3: display (first);  
break;  
default: exit (0);  
break;  
}
```

```
}  
case 2: printf("QUEUE\n");  
for (; ;)
```

```
{ printf("\n1: Insert-rear\n2: Delete-front\n3: Display-list\n4: Exit\n");  
printf("Enter the choice\n");  
scanf("%d", &choice);  
switch (choice)
```

```
{  
case 1: printf("Enter the item at rear-end\n");  
scanf("%d", &item);  
first = insert-rear (first, item);  
break;  
case 2: first = delete-front (first);  
break;
```

```
case 3: display(first);  
        break;
```

```
default: exit(0);
```

```
}
```

```
}
```

```
case 3: exit(0);
```

```
default: printf("Invalid choice\n");
```

```
}
```

```
}
```

```
}
```