9/12/2020

LP# 8

WAP to implement Stack and Queues 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 (size of (struct node));
if (x == NULL)
{ printf("memo full \n");
exit(0);
}
return x;
}
void freenode (NODE x) {
free(x); }
```

```
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_rear (NODE first) {
  NODE cur, prev;
  if (first == NULL) {
  printf ("stack 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\n", cur -> info);
free (cur);
prev -> link = NULL;
return first;
}


NODE delete_front (NODE first) {
NODE temp;
if (first == NULL) {
printf ("~~list~~ queue 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;
}
```

```
void display (NODE first) {
NODE temp;
if (first == NULL)
printf ("list empty cannot display items \n");
for (temp = first; temp != NULL; temp = temp -> link) {
printf ("%d\n", temp -> info);
}
}


int main () {
int item, choice;
 NODE first = NULL;
 printf ("STACK - insert rear and delete rear \n QUEUE - insert rear and
  delete front \n");
 printf ("\n1: Insert rear \n 2: delete ~~front~~ rear \n 3: delete front \n
  4: display_list \n 5: Exit\n");
  do
  {
 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 : first = delete_front (first);
         break;
case 4:  display (first);
         break;
case 5 : break;
default : printf ("invalid choice");
         break;
 }
} while (choice != 5);
return 0;
}
```